A Deep Learning Pipeline for Image Understanding and Acoustic Modeling

by

Pierre Sermanet

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

January 2014

Professor Yann LeCun

Dedication

To my family.

Acknowledgements

I would like to thank my advisor Prof. Yann LeCun for his insightful guidance and for giving me a chance to work on many wonderful projects before and during my thesis. Many thanks to Rob Fergus and David Eigen for sharing numerous late nights of work and intense collaboration. Many thanks to Raia Hadsell and Jan Ben for sharing countless freezing days of robotics development in the New Jersey outdoors and Urs Muller for trusting me with this very special project. I would like to thank Brian Kingsbury for all his support during our collaboration with IBM. Thanks to Matthieu Devin and Andrew Ng for hosting me at Google, it was a great pleasure to work with the Brain group. I am grateful for having worked with great people at NYU: Soumith Chintala, Koray Kavukcuoglu, Clement Farabet, Ylan Boureau, Michael Mathieu, Marco Scoffier, Ayse Erkan, Matt Grimes, Marc-Aurelio Ranzato, Li Wan and Xiang Zhang. Special thanks to Nathan Silberman for providing French impressions and Mr Goodbar. Finally, I would like to thank my family and friends for their support and encouragement during this long journey.

This effort uses the IARPA Babel Program Cantonese language collection release IARPA-babel101b-v0.4c and Vietnamese language collection release release babel107b-v0.7 full language packs.

Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

Abstract

One of the biggest challenges artificial intelligence faces is making sense of the real world through sensory signals such as audio or video. Noisy inputs, varying object viewpoints, deformations and lighting conditions turn it into a high-dimensional problem which cannot be efficiently solved without learning from data. This thesis explores a general way of learning from high dimensional data (video, images, audio, text, financial data, etc.) called deep learning. It strives on the increasingly large amounts of data available to learn robust and invariant internal features in a hierarchical manner directly from the raw signals. We propose an unified pipeline for feature learning, recognition, localization and detection using Convolutional Networks (ConvNets) that can obtain state-of-the-art accuracy on a number of pattern recognition tasks, including acoustic modeling for speech recognition and object recognition in computer vision. ConvNets are particularly well suited for learning from continuous signals in terms of both accuracy and efficiency. Additionally, a novel and general deep learning approach to detection is proposed and successfully demonstrated on the most challenging vision datasets. We then generalize it to other modalities such as speech data. This approach allows accurate localization and detection objects in images or phones in voice signals by learning to predict boundaries from internal representations. We extend the reach of deep learning from classification to detection tasks in an integrated fashion by learning multiple tasks using a single deep model. This work is among the first to outperform human vision and establishes a new state of the art on some computer vision and speech recognition benchmarks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Towards the end of the last century, machine intelligence reached high above human intelligence but only for highly specific and rigid tasks. Computers have beaten the brightest human chess players while being utterly incapable to converse with them or recognize objects in the scene. Such tasks seem trivial to humans because they learn from birth to interpret the highly complex and dynamic world they evolve in. Computer programs however reside in deterministic and fixed worlds. Making sense of the world through sensory signals such as audio or visual signals is too high dimensional a task to be programmed by humans. This work aims to provide machines the ability to understand and interact with the real world, by learning from data.

In this thesis, we explore a general way of learning from high dimensional data (video, images, audio, text, financial data, etc.) called deep learning. It strives on the increasingly large amounts of data available to learn robust and invariant internal features in a hierarchical manner, directly from raw signals. These representations, invariant to input changes such as noise, viewpoints, translation, rotations, scaling, deformations or lighting are the gateway from real-world noisy data to fixed codes that machines can

interpret. We explore different ways to learn rich feature representations, and use these to address several problems for different modalities. In particular, we tackle the following tasks by increasing order of difficulty: classification, localization and detection, each task being a sub-task of the next.

Figure 1.1: **Classification example for ImageNet LSVRC13.** This validation image contains one main object with groundtruth "pencil sharpener". Our model returns 5 guesses ordered by decreasing confidence. The classification is considered correct if one of the 5 guesses matches the groundtruth.

1.2 Problems

Throughout the thesis, we report results on a wide range of renowned datasets. Each problem is evaluated as follow. For classification tasks (e.g. Figure 1.1 and Figure 1.2), a model must predict the correct label of an image or phone. In the case of the 2013 ImageNet Large Scale Visual Recognition Challenge (ILSVRC13), up to five guesses are allowed to predict the correct answer because images can contain multiple unlabeled objects. The localization task (e.g. Figure 1.3) is similar to classification in that 5 guesses

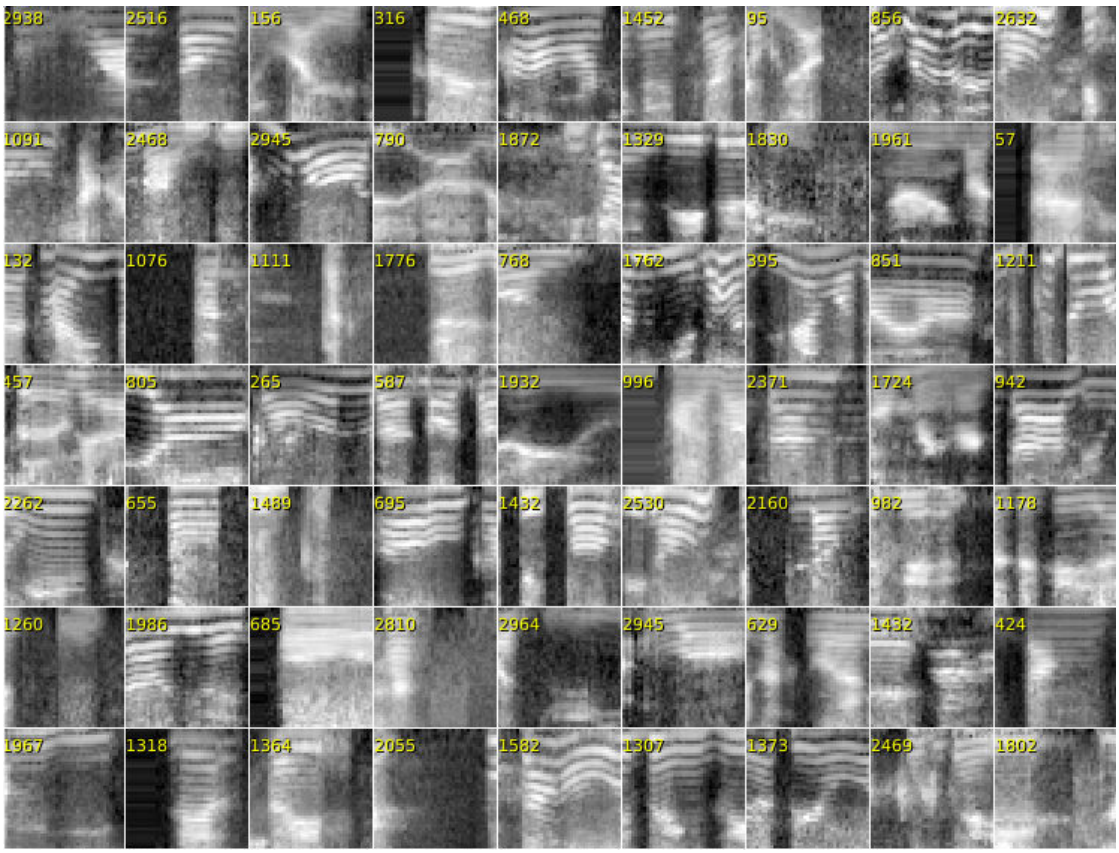

Figure 1.2: **Classification example for Babel Cantonese phones.** Sample labels range from 1 to 3000. The vertical axis is the frequency dimension (40 log-mel features) The horizontal axis the time dimension. Here, the time window is 41, the central column corresponding to the sample label. These time-frequency representations can be analyzed the same way as images.

are allowed per image. But additionally, a bounding box of the main object must be returned and must match with the groundtruth by 50% (using the PASCAL criterion of union over intersection). Each returned bounding box must be labeled with the correct class, i.e. bounding boxes and labels are not dissociated. Detection tasks (e.g. Figure 1.4) differ from localization in that there can be any number of objects in each image (including zero), and that false positives are penalized by the mean average precision (mAP) measure. The localization task is a convenient intermediate step between classification and detection in order to evaluate a localization method independently of

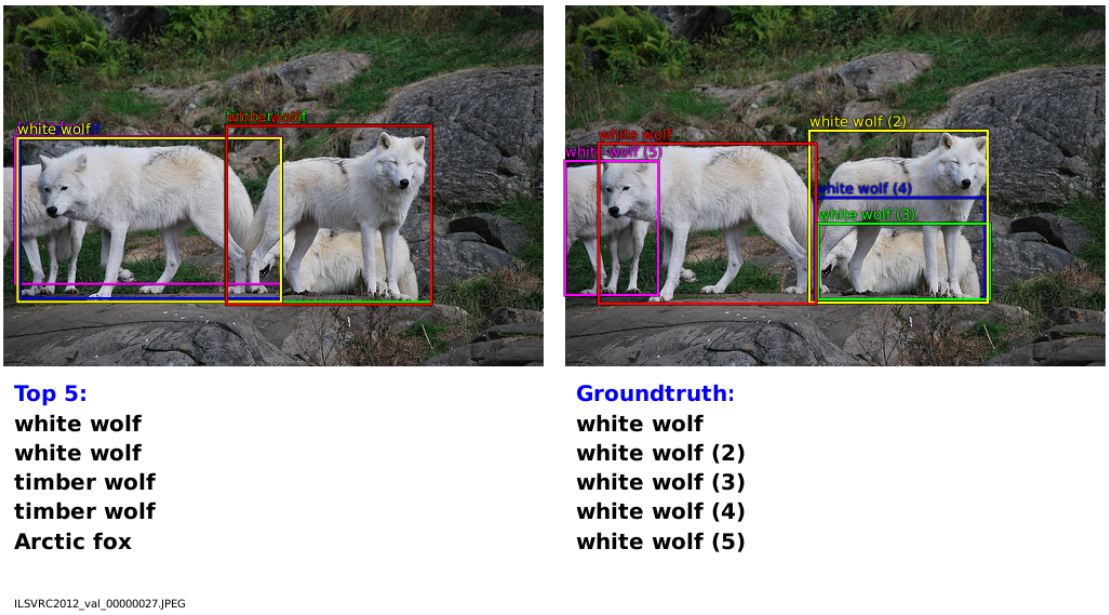

Figure 1.3: **Localization example for ImageNet LSVRC13.** The left image contains our predictions (ordered by decreasing confidence) while the right image shows the groundtruth labels. The localization is considered correct if one of the 5 guesses matches one of the groundtruth answer for both its class and its bounding box (at least 50% of the intersection over the union).

challenges specific to detection (such as learning a background class for instance).

Figure 1.4: **Detection example for ImageNet LSVRC13.** The left image contains our predictions (ordered by decreasing confidence) while the right image shows the groundtruth labels. This example illustrates the higher difficulty of the detection dataset compared to the classification and localization data only. The detection image may contain many small objects while the classification and localization images typically contain a single large object. Performance is measured using the mean average precision (mAP). Correct answers must match the groundtruth's class and bounding box, other answers count as false positives.

1.3 Summary of Contributions

We summarize here the main contributions of this thesis. Related work is reviewed in Chapter 2 and conclusions and directions for future work are addressed in Chapter 5.

1. **We present a novel deep learning approach to object detection which yields world record accuracy on the 2013 ImageNet Large Scale Visual Recognition Challenge (ILSVRC13) localization and detection datasets.** Using a shared feature learning pipeline with a classifier, we learn to predict object bounding boxes. We then accumulate many bounding box predictions and fuse them into single predictions. This method can handle any bounding box aspect ratio. It increases localization accuracy and robustness to false positives over traditional non-maximum suppression. We also suggest that by combining many localization predictions, detection can be performed without training on background samples and that it is possible to avoid the time-consuming and complicated bootstrapping training passes. Not training on background also lets the network focus solely on positive classes for higher accuracy.

2. **We established the first superhuman visual pattern recognition in an official international competition (test set known only to the organizers) along with [1].** During phase I of the 2011 German Traffic Sign Recognition Benchmark challenge, we pushed classification accuracy of traffic sign images above human performance (98.81%) with 98.98% accuracy, by improving on the traditional ConvNet architecture. One of the main improvements was the use of multi-stage features as input to the classifier as opposed to using the last feature layer only.

3. **We show that a single feature pipeline shared across multiple tasks can yield competitive or state of the art results.** On the ILSVRC13 data, features were initially learned by training on the classification task, and later reused for

the localization and detection tasks. Classification results were very competitive (13.6% error) and localization and detection ranked first against all other teams.

4. **We show that a single learning framework can be successfully applied to different modalities.** After obtaining state of the art results on vision datasets, we applied the same methods to speech recognition data and observed improvements over the baseline results.

5. **We demonstrate that unsupervised deep learning can significantly boost performance and obtained state of the art results for pedestrian detection.**

6. **We study and establish a series of best practices for the use of ConvNets for classification, localization and detection problems and propose a few important twists which consistently yield state of the art and competitive results on a range of classification and detection benchmarks.**

Chapter 2

Literature Survey

2.1 Feature Learning

Until recently, many state-of-the-art computer vision methods use a combination of hand-crafted features such as *Integral Channel Features* [2], HoG [3] and their variations [4, 5] and combinations [6], followed by a trainable classifier such as SVM [4, 7], boosted classifiers [2] or random forests [8]. While low-level features can be designed by hand with good success, mid-level features that combine low-level features are difficult to engineer without the help of some sort of learning procedure. Multi-stage recognizers that learn hierarchies of features tuned to the task at hand can be trained end-to-end with little prior knowledge (see an example of low-level features trained with unsupervised learning in Figure 2.1). Convolutional Networks (ConvNets) [9] are examples of such hierarchical systems with end-to-end feature learning that are trained in a supervised fashion. Recent works have demonstrated the usefulness of unsupervised pre-training for end-to-end training of deep multi-stage architectures using a variety of techniques such as stacked restricted Boltzmann machines [10], stacked auto-encoders [11] and stacked sparse auto-encoders [12], and using new types of non-linear transforms at each layer [13, 14].

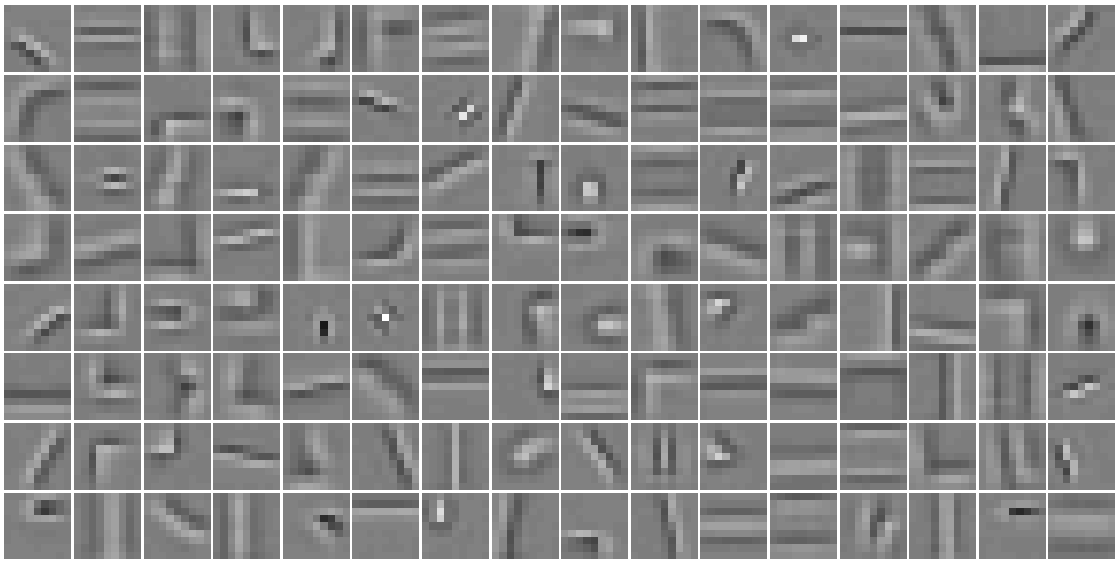

Figure 2.1: 128 9×9 filters trained on grayscale INRIA pedestrian images using ConvPSD [14]. It can be seen that in addition to edge detectors at multiple orientations, our system also learns more complicated features such as corner and junction detectors.

Recognizing the category of the dominant object in an image is a task to which Convolutional Networks (ConvNets) [15] have been applied for many years, whether the objects were handwritten characters [16], house numbers [17], textureless toys [18], traffic signs [19, 20], objects from the Caltech-101 dataset [13], or objects from the 1000-category ImageNet dataset [21]. The accuracy of ConvNets on small datasets such as Caltech-101, while decent, has not been record-breaking. However, the advent of larger datasets has enabled ConvNets to significantly advance the state of the art on datasets such as the 1000-category ImageNet [22].

The main advantage of ConvNets for many such tasks is that the entire system is trained *end to end*, from raw pixels to ultimate categories, thereby alleviating the requirement to manually design a suitable feature extractor. The main disadvantage is their ravenous appetite for labeled training samples.

For these reasons, and because labeled data has become increasingly available, Deep Neural Networks (DNN) have also yielded large improvements in the domain of speech

recognition [23]. In turn, ConvNets have advanced the state of the art over DNNs [24, 25]. The line between the field of computer vision and speech recognition is becoming increasingly blurry, techniques that have existed for many years in one field are now applied to the other field. In this thesis, we attempt to transfer more of the computer vision techniques and reinforce the bridge to speech recognition.

2.2 Object Detection

Many authors have proposed to use ConvNets for detection and localization with a sliding window over multiple scales, going back to the early 1990's for multi-character strings [26], faces [27], and hands [28]. More recently, ConvNets have been shown to yield state of the art performance on text detection in natural images [29], face detection [30, 31] and pedestrian detection [32].

Several authors have also proposed to train ConvNets to directly predict the instantiation parameters of the objects to be located, such as the position relative to the viewing window, or the pose of the object. For example Osadchy *et al.* [31] describe a ConvNet for simultaneous face detection and pose estimation. Faces are represented by a 3D manifold in the nine-dimensional output space. Positions on the manifold indicate the pose (pitch, yaw, and roll). When the training image is a face, the network is trained to produce a point on the manifold at the location of the known pose. If the image is not a face, the output is pushed away from the manifold. At test time, the distance to the manifold indicates whether the image contains a face, and the position of the closest point on the manifold indicates pose. Taylor *et al.* [33, 34] use a ConvNet to estimate the location of body parts (hands, head, etc) so as to derive the human body pose. They use a metric learning criterion to train the network to produce points on a body pose manifold. Hinton et al. have also proposed to train networks to compute explicit instantiation parameters of features as part of a recognition process [35].

Other authors have proposed to perform object localization via ConvNet-based seg-

mentation. The simplest approach consists in training the ConvNet to classify the central pixel (or voxel for volumetric images) of its viewing window as a boundary between regions or not [36]. But when the regions must be categorized, it is preferable to perform *semantic segmentation*. The main idea is to train the ConvNet to classify the central pixel of the viewing window with the category of the object it belongs to, using the window as context for the decision. Applications range from biological image analysis [37], to obstacle tagging for mobile robots [38] to tagging of photos [39]. The advantage of this approach is that the bounding contours need not be rectangles, and the regions need not be well-circumscribed objects. The disadvantage is that it requires dense pixel-level labels for training. This segmentation pre-processing or object proposal step has recently gained popularity in traditional computer vision to reduce the search space of position, scale and aspect ratio for detection [40, 41, 42, 43]. Hence an expensive classification method can be applied at the optimal location in the search space, thus increasing recognition accuracy. Additionally, [43, 44] suggest that these methods improve accuracy by drastically reducing unlikely object regions, hence reducing potential false positives. Recently, [45] have established a new state of the art on the PASCAL dataset by using a ConvNet to classify object proposals. Our dense sliding window method however outperforms object proposal methods on the ILSVRC13 detection dataset.

Krizhevsky *et al.* [21] demonstrated impressive localization performance using a large ConvNet during the ImageNet 2012 competition. There has been however no published work describing their approach. We are thus the first to provide a clear explanation how ConvNets can be used for localization and detection for ImageNet data.

In this paper we use the terms localization and detection in a way that is consistent with their use in the ImageNet 2013 competition, namely that the only difference is the evaluation criterion used and both involve predicting the bounding box for each object in the image.

Chapter 3

Feature Learning

In this chapter, we explore Convolutional Network (ConvNet) architectures for classification of visual and audio signals. A range of architectural enhancements are successfully applied to a number of vision tasks, yielding accuracy records on international classification datasets and challenges, including the Street-View House Numbers dataset (SVHN, Figure 3.1), the German Traffic Sign Recognition Benchmark (GTSRB, Figure 3.2) and the Imagenet Large Scale Visual Recognition Challenge 2013 (ILSVRC13).

3.1 ConvNet Architectures for Computer Vision

In this section, we experiment with the most trendy ConvNet architecture recently introduced by Krizhevsky *et al.* [21]. However we improve on the network design and the inference step. Because of time constraints, some of the training features in Krizhevsky's model were not explored, it is thus expected that results can be improved even further. These are discussed in Chapter 5.

3.1.1 Model Design and Training

We train the network on the ImageNet 2012 training set (1.2 million images and $C = 1000$ classes) [22]. Our model uses the same fixed input size approach proposed

Figure 3.1: **The StreetView House Numbers dataset (SVHN)** by [46]. This classification dataset contains approximately 600,000 colored samples of size 32x32 distributed among 10 digit classes. The task is to classify the digit at the center of the patch. Additional digits may be present on either sides.

by Krizhevsky *et al.* [21] during training but turns to multi-scale for classification as described in the next section. Each image is downsampled so that the smallest dimension is 256 pixels. We then extract 5 random crops (and their horizontal flips) of size 221x221 pixels and present these to the network in mini-batches of size 128. The weights in the network are initialized randomly with $(\mu, \sigma) = (0, 1 \times 10^{-2})$. They are then updated by stochastic gradient descent, accompanied by momentum term of 0.6 and an ℓ_2 weight decay of 1×10^{-5}. The learning rate is initially 5×10^{-2} and is successively decreased by a factor of 0.5 after $(30, 50, 60, 70, 80)$ epochs. DropOut [48] with a rate of 0.5 is employed on the fully connected layers (6th and 7th) in the classifier.

We detail the architecture sizes in tables 3.1 and 3.2. Note that during training, we treat this architecture as non-spatial (output maps of size 1x1) as opposed to the

Figure 3.2: **The German Traffic Sign Recognition Benchmark (GTSRB)** by [47]. This dataset contains 43 classes and 26,640 training samples. These samples are among the most difficult ones to classify because of challenging real-world variations such as viewpoints, lighting conditions (saturations, low-contrast), motion-blur, occlusions, sun glare, physical damage, colors fading, graffiti, stickers and an input resolutions as low as 15x15.

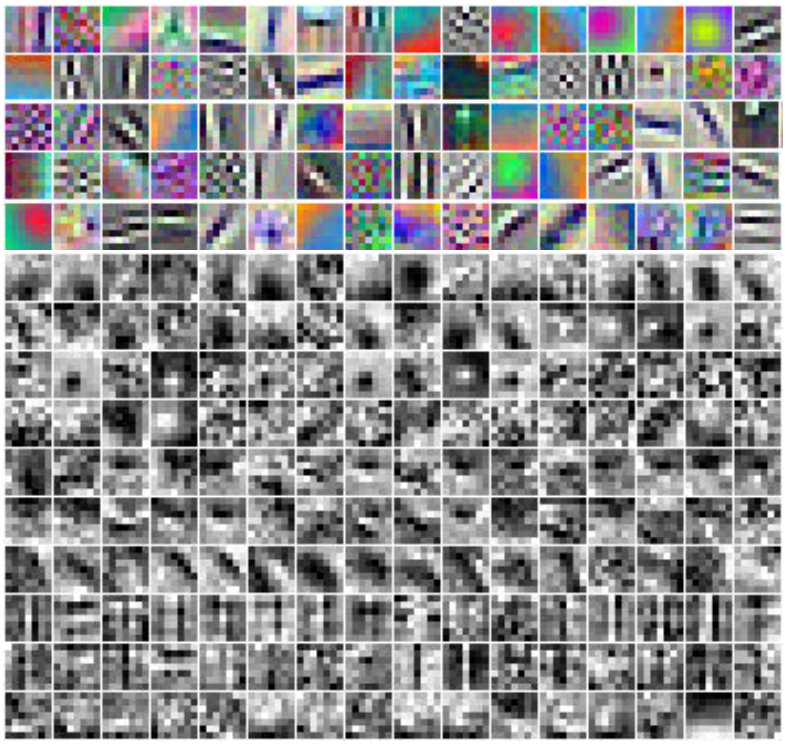

Figure 3.3: **Layer 1 (top) and layer 2 filters (bottom).**

inference step which produces spatial outputs. Layers 1-5 are similar to Krizhevsky *et al.* [21], using rectification ("*relu*") non-linearities and max pooling, but with the

Layer	1	2	3	4	5	6	7	Output 8
Stage	conv + max	conv + max	conv	conv	conv + max	full	full	full
# channels	96	256	512	1024	1024	3072	4096	1000
Filter size	11x11	5x5	3x3	3x3	3x3	-	-	-
Conv. stride	4x4	1x1	1x1	1x1	1x1	-	-	-
Pooling size	2x2	2x2	-	-	2x2	-	-	-
Pooling stride	2x2	2x2	-	-	2x2	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	231x231	24x24	12x12	12x12	12x12	6x6	1x1	1x1

Table 3.1: **Architecture specifics for model A (or "fast" model).** The spatial size of the feature maps depends on the input image size, which varies during our inference step – see Table 3.4. Here we show training spatial sizes. Note that layer 5 is the top convolutional layer, with subsequent layers being fully connected, being used a classifier which is applied in sliding window fashion to the layer 5 maps. These fully-connected layers can be seen as 1x1 convolutions in a spatial setting.

Layer	1	2	3	4	5	6	7	8	Output 9
Stage	conv + max	conv + max	conv	conv	conv	conv + max	full	full	full
# channels	96	256	512	512	1024	1024	4096	4096	1000
Filter size	7x7	7x7	3x3	3x3	3x3	3x3	-	-	-
Conv. stride	2x2	1x1	1x1	1x1	1x1	1x1	-	-	-
Pooling size	3x3	2x2	-	-	-	3x3	-	-	-
Pooling stride	3x3	2x2	-	-	-	3x3	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	221x221	36x36	15x15	15x15	15x15	15x15	5x5	1x1	1x1

Table 3.2: **Architecture specifics for model B (or "slow" model).** It differs from the model A mainly in the stride of the first convolution, the number of stages and the number of feature maps.

model	# parameters (in millions)	# connections (in millions)
Krizhevsky	60	-
A	145	2810
B	144	5369

Table 3.3: **Number of parameters and connections** for different models.

following differences: (i) no contrast normalization is used; (ii) pooling regions are non-overlapping and (iii) our model has larger 1st and 2nd layer feature maps, thanks to a smaller stride (2 instead of 4). A larger stride is beneficial for speed but will hurt

accuracy.

In Figure 3.3, we show the filter coefficients from the first two convolutional layers. The first layer filters capture orientated edges, patterns and blobs. In the second layer, the filters have a variety of forms, some diffuse, others with strong line structures or oriented edges.

3.1.2 Feature Extractor

Along with this work, we have releases a feature extractor dubbed "OverFeat" [1] in order to provide powerful features for computer vision research. Two models are provided, a *fast* and *slow* one. Each architecture is described in tables 3.1 and 3.2. We also compare their sizes in Table 3.3 in terms of parameters and connections. The *slow* model is more accurate than the *fast* one (14.18% classification error as opposed to 16.39% in Table 3.5), however it requires nearly twice as many connections. Using a committee of 7 *slow* models reaches 13.6% classification error as shown in Figure 3.5.

3.1.3 Multi-Scale Classification

In [21], multi-view voting is used to boost performance: a fixed set of 10 views (4 corners and center, with horizontal flip) is averaged. Not only may this approach ignore some regions of the image, it may also be computationally redundant if views overlap. Additionally, it is only applied at a single scale, which may not be the scale at which the ConvNet will respond with optimal confidence. Instead, we explore the entire image by densely running the network at each location and at multiple scales. While the sliding window approach may be computationally prohibitive for certain types of model, it is inherently efficient in the case of ConvNets (see section 4.2). This approach yields significantly more views for voting, which increases robustness while remaining computationally efficient. The result of convolving a ConvNet on an image of arbitrary

[1] http://cilvr.nyu.edu/doku.php?id=software:overfeat:start

size is a spatial map of C-dimensional vectors at each scale.

The total subsampling ratio in the network described above is 2x3x2x3, or 36. Hence when applied densely, this architecture can only produce a classification vector every 36 pixels in the input dimension along each axis. This coarse distribution of outputs decreases performance compared to the 10-view scheme because the network windows are not well aligned with the objects in the images. The better aligned the network window and the object, the strongest the confidence of the network response. To circumvent this problem, we take the approach introduced by Giusti *et al.* [49] by avoiding the last subsampling operation (x3), yielding a subsampling ratio of x12 instead of x36.

We now explain in details how the resolution augmentation is performed. We use 6 scales of input which result in unpooled layer 5 maps of varying resolution (see Table 3.4 for details). These are then pooled and presented to the classifier using the following procedure, which is accompanied by Figure 3.4:

(a) For a single image, at a given scale, we start with the unpooled layer 5 feature maps.

(b) Each of unpooled maps undergoes a 3x3 max pooling operation (non-overlapping regions), repeated 3x3 times for (Δ_x, Δ_y) pixel offsets of $\{0, 1, 2\}$.

(c) This produces a set of pooled feature maps, replicated (3x3) times for different (Δ_x, Δ_y) combinations.

(d) The classifier (layers 6,7,8) has a fixed input size of 5x5 and produces a C-dimensional output vector for each location within the pooled maps. The classifier is applied in sliding-window fashion to the pooled maps, yielding C-dimensional output maps (for a given (Δ_x, Δ_y) combination).

(e) The output maps for different (Δ_x, Δ_y) combinations are reshaped into a single 3D output map (two spatial dimensions x C classes).

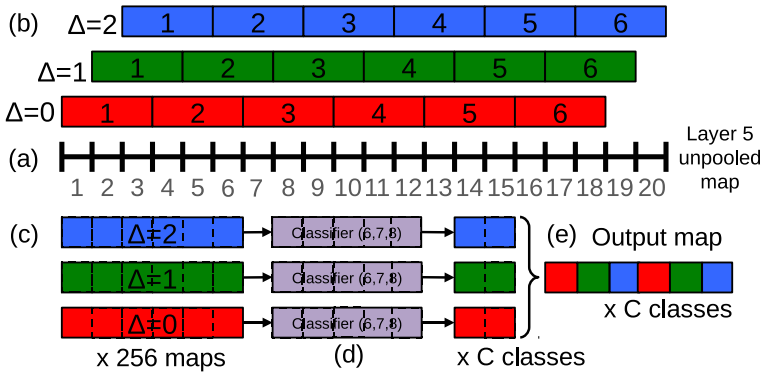

Figure 3.4: 1D illustration (to scale) of output map computation for classification, using y-dimension from scale 2 as an example (see Table 3.4). (a): 20 pixel unpooled layer 5 feature map. (b): max pooling over non-overlapping 3 pixel groups, using offsets of $\Delta = \{0, 1, 2\}$ pixels (red, green, blue respectively). (c): The resulting 6 pixel pooled maps, for different Δ. (d): 5 pixel classifier (layers 6,7) is applied in sliding window fashion to pooled maps, yielding 2 pixel by C maps for each Δ. (e): reshaped into 6 pixel by C output maps.

Scale	Input size	Layer 5 pre-pool	Layer 5 post-pool	Classifier map (pre-reshape)	Classifier map size
1	245x245	17x17	(5x5)x(3x3)	(1x1)x(3x3)xC	3x3xC
2	281x317	20x23	(6x7)x(3x3)	(2x3)x(3x3)xC	6x9xC
3	317x389	23x29	(7x9)x(3x3)	(3x5)x(3x3)xC	9x15xC
4	389x461	29x35	(9x11)x(3x3)	(5x7)x(3x3)xC	15x21xC
5	425x497	32x35	(10x11)x(3x3)	(6x7)x(3x3)xC	18x24xC
6	461x569	35x44	(11x14)x(3x3)	(7x10)x(3x3)xC	21x30xC

Table 3.4: **Spatial dimensions of our multi-scale approach**. 6 different sizes of input images are used, resulting in layer 5 unpooled feature maps of differing spatial resolution (although not indicated in the table, all have 256 feature channels). The (3x3) results from our dense pooling operation with $(\Delta_x, \Delta_y) = \{0, 1, 2\}$. See text and Figure 3.4 for details for how these are converted into output maps.

These operations can be viewed as shifting the classifier's viewing window by 1 pixel through pooling layers without subsampling and using skip-kernels in the following layer (where values in the neighborhood are non-adjacent).

The procedure above is repeated for the horizontally flipped version of each image. We then produce the final classification by (i) taking the spatial max for each class,

at each scale and flip; (ii) averaging the resulting C-dimensional vectors from different scales and flips and (iii) taking the top-1 or top-5 elements (depending on the evaluation criterion) from the mean class vector.

The scheme described above has several notable properties. First, the two halves of the network, i.e. the feature extraction layers (1-5) and classifier layers (6-output), are used in opposite ways. In the feature extraction portion, the filters are convolved across the entire image in one pass. For a computational perspective, this is far more efficient than sliding a fixed-size feature extractor over the image and then aggregating the results from different locations[2]. However, these principles are reversed for the classifier portion of the network. Here, we want to hunt for a fixed-size representation in the layer 5 feature maps across different positions and scales. Thus the classifier has a fixed-size 5x5 input and is exhaustively applied to the layer 5 maps. Second, the overlapping pooling scheme (with single pixel shifts (Δ_x, Δ_y)) ensures that we can obtain fine alignment between the classifier and the representation of the object in the feature map input. Third, our pooling scheme is similar to Giusti *et al.* [49] who shift the classifier's viewing window by 1 pixel through pooling layers without subsampling and use skip-kernels in the following layer (where values in the neighborhood are non-adjacent). Finally, the dense manner in which the classifier is applied also helps to improve performance. We explore this in Section 3.1.4, where we enable/disable the pixel shifts to reveal their performance contribution.

3.1.4 Results

In Table 3.5, we experiment with different approaches and for reference compare them to the single network model of Krizhevsky *et al.* [21]. The approach described above, with 6 scales, achieves a top-5 error rate of 13.6%. As might be expected, using fewer scales hurts performance, the single-scale model is worse with 16.97% top-5 error.

[2]Our network with 6 scales takes around 2 secs on a K20x GPU to process one image

The fine stride technique illustrated in Figure 3.4 brings a relatively small improvement in the single scale regime, but is also of importance for the multi-scale gains shown here.

Approach	Top-1 error %	Top-5 error %
Krizhevsky *et al.* [21]	40.7	18.2
OverFeat - 1 fast model, 10 views	39.38	17.16
OverFeat - 1 fast model, scale 1, coarse stride	39.28	17.12
OverFeat - 1 fast model, scale 1, fine stride	39.01	16.97
OverFeat - 1 fast model, 4 scales (1,2,4,6), fine stride	38.57	16.39
OverFeat - 1 fast model, 6 scales (1-6), fine stride	38.12	16.27
OverFeat - 1 big model, 10 views	35.60	14.71
OverFeat - 1 big model, 4 scales, fine stride	35.74	14.18
OverFeat - 7 fast models, 4 scales, fine stride	35.10	13.86
OverFeat - 7 big models, 4 scales, fine stride	33.96	13.24

Table 3.5: **Classification experiments on validation set.** Fine/coarse stride refers to the number of Δ values used when applying the classifier. Fine: $\Delta = 0, 1, 2$; coarse: $\Delta = 0$. 10 views refers to the multi-view scheme employed by [21], i.e. for each flip, average the 4 corners and center views.

We report the test set results of the 2013 competition in Figure 3.5 where our model (OverFeat) obtained 14.2% accuracy by voting of 7 ConvNets (each trained with different initializations) and ranked 5th out of 18 teams. The best accuracy using ILSVRC13 data only was 11.7%. Pre-training with extra data from the ImageNet Fall11 dataset improved this number to 11.2%. In post-competition work, we improve the OverFeat results down to 13.6% error by using bigger models (more features and more layers). Due to time constraints, these bigger models are not fully trained, more improvements are expected to appear in time.

3.2 ConvNet Architectures for Acoustic Modeling

This section illustrates how ConvNets can yield state of the art results not only on computer vision tasks but also on other modalities that exhibit local coherence such as audio power spectra.

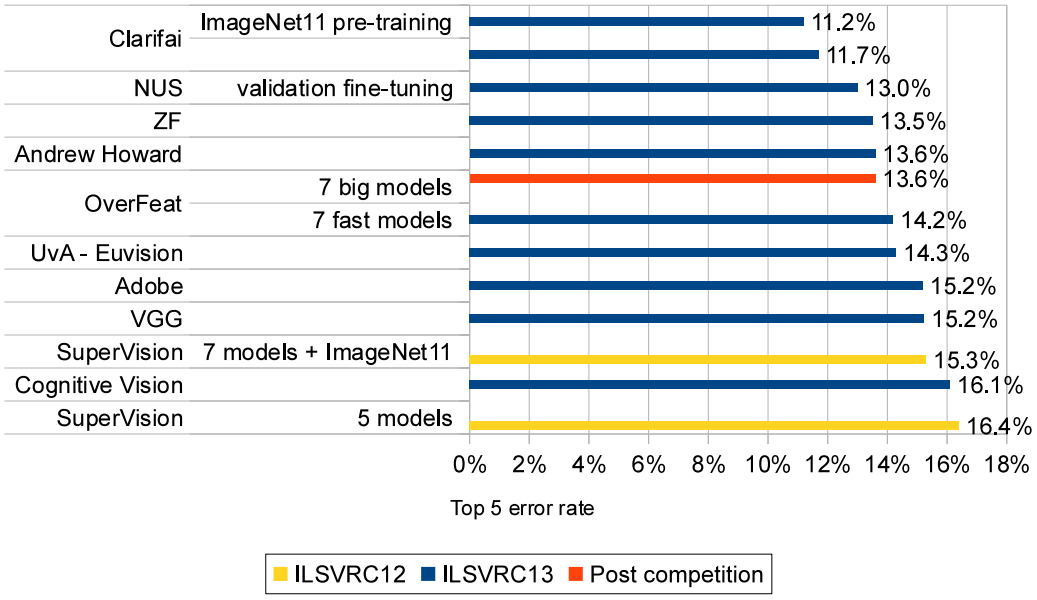

Figure 3.5: **Test set classification results.** During the competition, OverFeat yielded 14.2% top 5 error rate using an average of 7 fast models. In post-competition work, OverFeat ranks fourth with 13.6% error using bigger models (more features and more layers).

3.2.1 Architecture

We reuse the OverFeat framework that we applied to vision datasets and adapt the architecture for the IARPA Babel competition data. In this case, the network is not trained directly from the raw data but instead requires some pre-processing. Audio signals are first turned into a 2-dimensional signal, composed of a frequency dimension (log Mel features [50] in this case) and a temporal dimension. This 2D signal can be viewed as an image (see Figure 1.2) and treated as such by the learning pipeline from that point on. In future work, we think that a constant number of bands per octave (e.g. Constant Q) will be more suitable than the Mel scale for weight sharing across frequency. While using hand-designed features for pre-processing is currently the norm in speech recognition, [51] have recently obtained good results by learning a phoneme

sequence recognizer directly from the raw signal using ConvNets.

The architecture is very similar to the one we use for images, in that it uses a number of convolutional stages followed by fully connected layers with dropout regularization. It also does not use any layer normalization such as Local Contrast Normalization. It is however not as deep because the complexity of the problem and the input resolution are lower than that of ImageNet. The architecture is inspired by the best architecture found by Sainath *et al.* [24] which uses 2 convolutional stages and 4 fully connected layers. In this architecture (see Table 3.6), Max-Pooling is applied at the first stage only and solely along the frequency dimension. Experimentation by [24] suggests that temporal pooling does not improve results.

Model	Layer	deltas	1	2	3	4	5	Output 6
	Stage		conv + max pooling	conv	full	full	full	full
	Kernel size		9x9	4x4	-	-	-	-
	Conv. stride		1x1	1x1	-	-	-	-
	Pool. size		4x1	-	-	-	-	-
	Pool. stride		4x1	-	-	-	-	-
	Spatial input		40x41	8x33	5x30	1x1	1x1	1x1
A	# features	✓	64	64	1024	1024	1024	3000
	Dropout		-	-	-			✓
B	# features	✗	64	64	1024	1024	1024	3000
	Dropout		-	-	-	✓	✓	✓
C	# features	✗	64	64	4096	4096	4096	3000
	Dropout		-	-	✓	✓	✓	✓

Table 3.6: **Architecture specifics for our speech classification model.** Dropout has traditionally been used on the last 2 layers only, we found however that more dropout enforced a stronger regularization and improved results (see Table 3.7).

The main differences of our model compared to [24]'s model are:

1. **Class-balanced training.** The class-distributions in some datasets can be very unbalanced. Some classes have many samples while the rare ones will only have a few. Training can be performed in a class-equalized way, training on the same

number of sample for each class regardless of their true distribution, or in an unbalanced way using the natural sample distribution. Similarly, the inference step can be designed to perform in a balanced or unbalanced regime depending on the application. For example, in image segmentation, an unbalanced error measure will favor classes that occur frequently in natural images (sky, buildings, roads, etc.) but won't penalize mistakes on rare and small objects (pedestrians, traffic signs, etc.). For some applications, it might be more interesting to perform a little bit worse on frequent classes in order to correctly detect rare classes. Speech recognition favors an unbalanced inference regime. However, we argue that regardless what the desired inference regime is, one should initially train in a class-balanced fashion in order to learn the most general features possible. The rationale is that the more data, the more ConvNets can generalize. Training in an unbalanced fashion is equivalent to reducing the dataset to its most common classes because the network will mostly be looking at homogeneous samples. Once generic features have been learned, one can perform an unbalanced training fine-tuning phase. We argue this partly explains the improvements seen over the baseline model in Table 3.7.

2. **No use of delta input channels: 1x40x41.** Traditionally, speech models are fed first and second temporal derivatives (called delta and delta-delta) of the input map in addition to the input map itself. This aims to provide richer features to the learning model. By visually inspecting these delta channels, we hypothesize that the extra information they provide is not significant and that the corresponding operations can be learned by the network. Hence, for simplicity and speed in our early experiment, we use only the input map itself. Therefore the input size is 1x40x41 rather than 3x40x41, where the dimensions are ordered as follow: input channels, frequency and time. Future experiments should determine if delta channels can be beneficial in our model.

3. **Dropout.** [24] does not mention the use of Dropout for regularization. In our early experiments, we used a single layer of Dropout but that proved to be insufficient to reduce overfitting on the training set. Later, we experimented with more Dropout and found that applying it to the last 3 fully connected layers yielded the best results, which were significantly stronger than with just 1 Dropout layer (see Table 3.7).

model	architecture	training distribution	language	data	cross-entropy loss	sub-phone error %	phone phone error %
IBM	DNN	U	Vietnamese	held.	1.86	-	-
IBM	DNN	U	Vietnamese	val.	2.26	47.30	-
OverFeat	model A	B + U	Vietnamese	val.	2.35	48.54	30.94
IBM	DNN	U	Cantonese	held.	1.73	37.78	-
OverFeat	model A	B	Cantonese	val.	3.80	75.20	
OverFeat	model B + longer training	B	Cantonese	val.	3.42	71.30	
OverFeat	model C + longer training	B	Cantonese	val.	3.05	67.74	-
OverFeat	model A	B + U	Cantonese	val.	1.79	36.91	23.63
OverFeat	model B + longer training	B + U	Cantonese	val.	1.51	35.85	22.54
OverFeat	model C + longer training	B + U	Cantonese	val.	**1.38**	**33.97**	-

Table 3.7: **Classification results on Babel Cantonese and Vietnamese speech data.** The reported rates are not class-balanced during inference, i.e. normalized per-class by the number of samples in each class. However we report the class distribution used during training, unbalanced (U), balanced (B) or a combination of both. Results are reported on two different datasets, the validation and held-out sets. In early experiments, we used Dropout on only 1 layer. We later found that applying Dropout to 3 layers was effective at preventing overfitting and yielded significantly stronger results.

3.2.2 Results

In Table 3.7, we report improvements brought by our architecture over the existing baseline by IBM. The OverFeat framework yields a cross-entropy loss of 1.51 and a sub-phone error rate of 35.85%, while the baseline obtained a 1.73 loss and 37.78% error. At the time of writing, the baseline validation results are not available, hence no direct comparison can be made between the OverFeat validation results and the IBM held-out results. However, the held-out results tend to be more optimistic than the validation

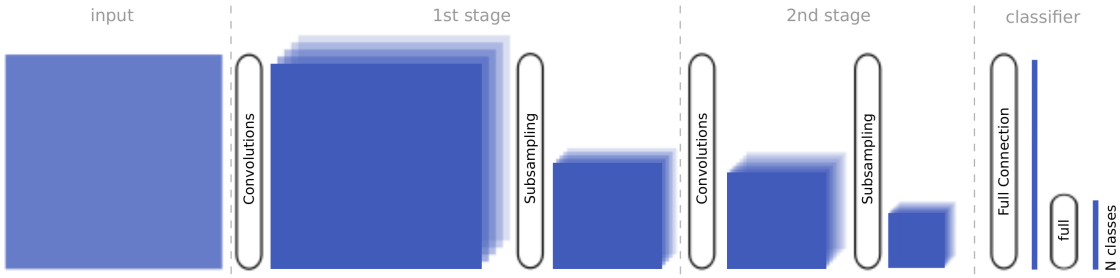

Figure 3.6: **A traditional Convolution Neural Network architecture**, composed of repeatable stages (two here) followed by a classifier. Each stage is mainly composed of a convolution and a subsampling layer, for more details see Figure 3.7. Classifiers can have one or multiple layers, the one depicted here has 2 fully-connected layers.

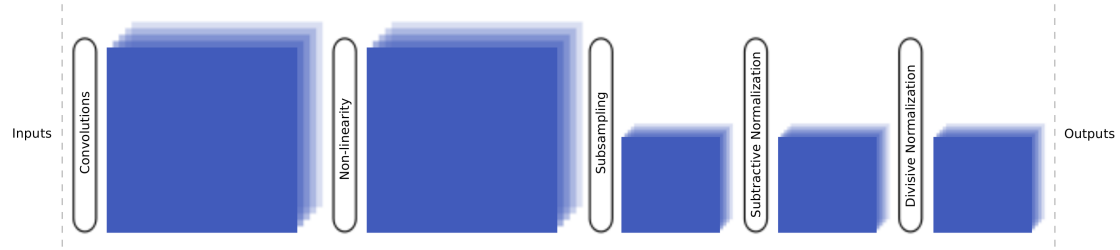

Figure 3.7: **A Convolutional Neural Network stage** starts with a convolution layer followed by a non-linearity such as a sigmoid function (e.g. tanh()). The pooling and subsampling then follows, e.g. a 2x2 L2 pooling and a 2x2 subsampling. Finally, normalization can optionally be used, either as a subtractive normalization only or subtractive and divisive normalizations (also called local constrast normalization). Stages can be repeated multiple times in a sequence, typically with more and more depth in feature space.

results, as shown by IBM's results on the Vietnamese data. OverFeat's comparative improvement is therefore expected to be greater than the current improvements.

3.3 ConvNets Enhancements

Convolutional Neural Networks (ConvNets) are traditionally composed of a sequence of repeatable stages followed by a classifier (Figure 3.6). Each stage is itself a sequence of layers, typically a convolutional layer, followed by a non-linearity layer, itself followed by a pooling and subsampling layer and sometimes ended by a normalization layer (See Figure 3.7). This stage architecture is repeated multiple times, twice for typical problems

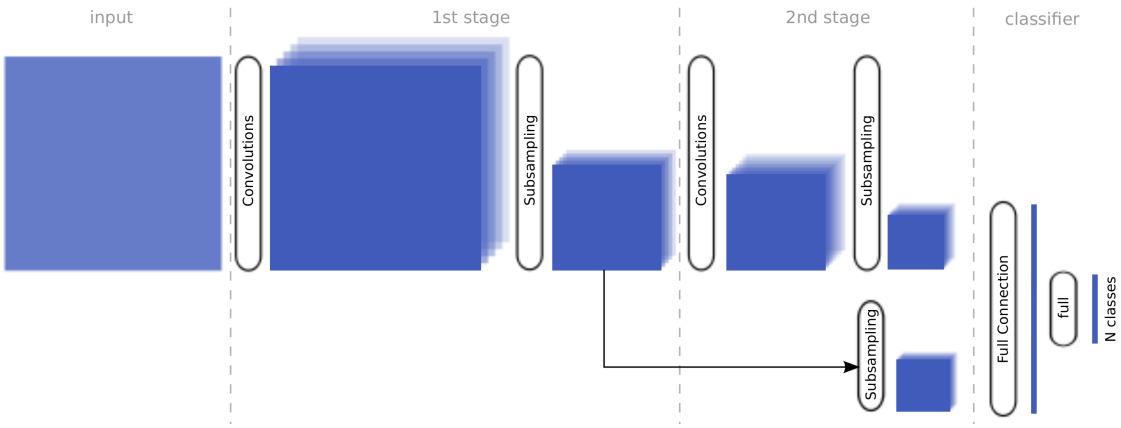

Figure 3.8: **A multi-stage ConvNet is a network** where features coming from multiple stages are concatenated together as input to the final classifier. This architecture can bring substantial accuracy improvements to complex vision tasks.

such as handwritten character classification [15] or more for larger problems [52].

3.3.1 Multi-Stage feature learning

The multi-stage (MS) features architecture differs from the traditional ConvNet in that the output of each stage is connected to the input of the classifier. Usual ConvNets are organized in strict feed-forward layered architectures in which the output of one layer is fed only to the layer above. Instead, the output of the first stage is branched out and fed to the classifier, in addition to the output of the second stage (Figure 3.8). Contrary to [53], we use the output of the first stage after pooling/subsampling rather than before. Additionally, applying a second subsampling stage on the branched output yields higher accuracies than without on a traffic sign classification task. Therefore the branched 1^{st}-stage outputs are more subsampled than in traditional ConvNets but overall undergoes the same amount of subsampling (4x4 here) than the 2^{nd}-stage outputs. The motivation for combining representation from multiple stages in the classifier is to provide different scales of receptive fields to the classifier. In the case of 2 stages of features, the second stage extracts "global" and invariant shapes and structures, while the first stage extracts "local" motifs with more precise details. This richer representation consistently improve

performance in a range of tasks [53, 20, 54, 17], from traffic sign and house numbers classification to pedestrian detection as reported in Table 3.8. While substantial gains are reported for pedestrian and traffic signs tasks, we only observe minimal gains on the house numbers dataset. Similarly, [25] applied this technique to speech recognition but observed very little gains while having to double the number of parameters. The likely explanation for this observation is that gains are correlated to the amount of texture and multi-scale characteristics of the objects of interest. Thus this method is not appropriate for all problems.

Task	Single-Stage features	Multi-Stage features	Improvement %
Pedestrians detection (INRIA) [54]	14.26%	9.85%	31%
Traffic Signs classification (GTSRB) [20]	1.80%	0.83%	54%
House Numbers classification (SVHN) [17]	5.54%	5.36%	3.2%

Table 3.8: Error rates improvements of multi-stage features over single-stage features for different types of objects detection and classification. Improvements are significant for multi-scale and textured objects such as traffic signs and pedestrians but minimal for house numbers.

3.3.2 Lp Pooling

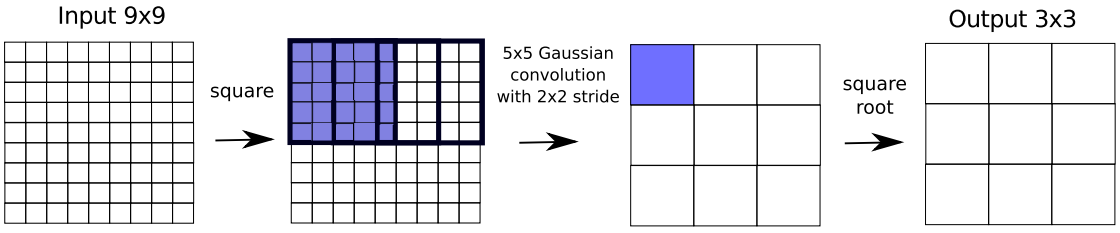

Figure 3.9: L2-pooling applied to a 9x9 feature map with a 3x3 Gaussian kernel and 2x2 stride

Lp pooling is a biologically inspired pooling layer modelled on complex cells [55, 56]

who's operation can be summarized in equation (1), where G is a Gaussian kernel, I is the input feature map and O is the output feature map. It can be imagined as giving an increased weight to stronger features and suppressing weaker features. Two special cases of Lp pooling are notable. $P = 1$ corresponds to a simple Gaussian averaging, whereas $P = \infty$ corresponds to max-pooling (i.e only the strongest signal is activated). Lp-pooling has been used previously in [57, 58] and a theoretical analysis of this method is described in [59].

$$O = (\sum \sum I(i,j)^P \times G(i,j))^{1/P} \tag{3.1}$$

Figure 3.9 demonstrates a simple example of L2-pooling.

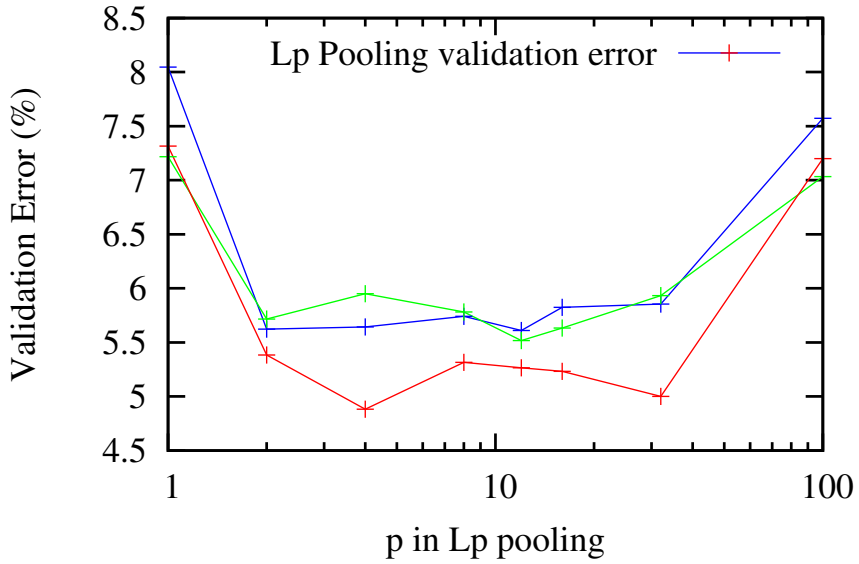

Figure 3.10: Error rate of Lp-pooling on 3 cross-validation sets for $p = 1, 2, 4, 8, 12, 16, 32, \infty$ ($p = \infty$) is represented as $p = 100$ for convenience). These validation errors are reported after 1000 training epochs.

For the pooling layers, we compare Lp-pooling for the value $p = 1, 2, 4, 8, 12, 16, 32, \infty$ on the validation set and use the best performing pooling on the final testing. The performance of different pooling methods on the validation set can be seen in Figure 3.10. Insights from [59] tell us that the optimal value of p varies for different input spaces and

there is no single globally optimal value for p. With validation data, we observe that $p = 2, 4, 12$ give the best performance ($5.62\%, 5.64\%$ and 5.61% respectively). Max-pooling ($p = \infty$) yielded a validation error rate of 7.57%.

Algorithm	SVHN-Test Accuracy
Binary Features (WDCH)	63.3%
HOG	85.0%
Stacked Sparse Auto-Encoders	89.7 %
K-Means	90.6%
ConvNet / MS / Average Pooling	**90.94%**
ConvNet / MS / L2 / Smaller training	**91.55%**
ConvNet / SS / L2	**94.46%**
ConvNet / MS / L2	**94.64%**
ConvNet / MS / L12	**94.89%**
ConvNet / MS / L4	**94.97%**
ConvNet / MS / L4 / Padded	**95.10%**
ConvNet / MS / L4 / RGB / Linear tanh	**95.72%**
Human Performance	98.0%

Table 3.9: Performance reported by [46] with the additional Supervised ConvNet models with state-of-the-art accuracy of 95.72%.

Our experiments demonstrate a clear advantage of Lp pooling with $1 < p < \infty$ on the house numbers dataset [46], in validation (Figure 3.10) and test (L2 pooling is 3.58 points superior to average pooling in Table 2). With L4 pooling, we obtain a state-of-the-art performance on the test set with an accuracy of 95.72% compared to the previous best accuracy of 90.6% (Table 3.9). Padding around inputs improves accuracy by 0.13% points from the 94.97% non-padded accuracy. This is likely explained by digit edges being very close to the image borders as seen in Figure 3.11. Padding allows centered edge filters to fire correctly at the borders.

Improvements to [17] bring the state of the art up to 95.72% accuracy by removing local contrast normalization on the input, a rather common procedure, and changing the nonlinearity to a linear tanh, as described in following sections. Minor architecture improvements are also reported and summarized in Figure 3.17.

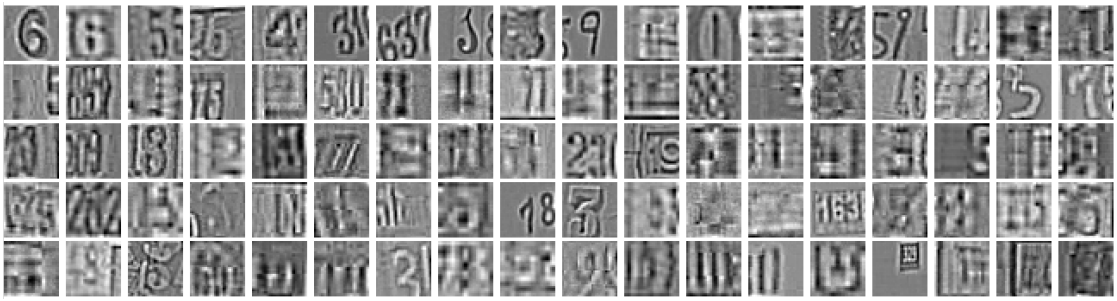

Figure 3.11: **Preprocessed Y channel of SVHN validation samples with highest energy** (i.e. highest error) with the 94.64% accuracy L2-pool based multi-stage ConvNet.

3.3.3 Preprocessing

Inputs are commonly preprocessed using local contrast normalization [13, 14, 20]. While preprocessing is usually beneficial in the presence for example of shadows in an image, we show that it significantly reduces performance on the house numbers classification task. In Figure 3.12, using RGB rather than preprocessed YUV consistently yielded around 1 point improvement in accuracy on validation data, regardless of the connection scheme on the input channels. This experiment suggests that the local contrast normalization can have a negative impact on some datasets. However, on others where lighting conditions greatly vary, normalization can be critical. An example of extreme lighting condition is demonstrated in Figure 3.13 where a sample taken from the GTSRB traffic sign dataset is uniformly black to the human eye. Normalization later clearly reveals a traffic sign. Although a ConvNet can detect small gradients while the human eye cannot, the resulting activations will be minimal and will not fall into the usual range of activations induced by most samples. Hence normalization facilitates learning of these extreme examples by shifting the activation ranges to normal ones. This particular sample is cropped out of an entire street scene with brighter pixels, in which a global normalization over the image would not be sufficient. The local contrast normalization however operates locally and allows proper normalization even if the rest of the image is within normal pixel ranges. This normalization was successfully applied

by [60, 61] to further improve their results on the GTSRB dataset.

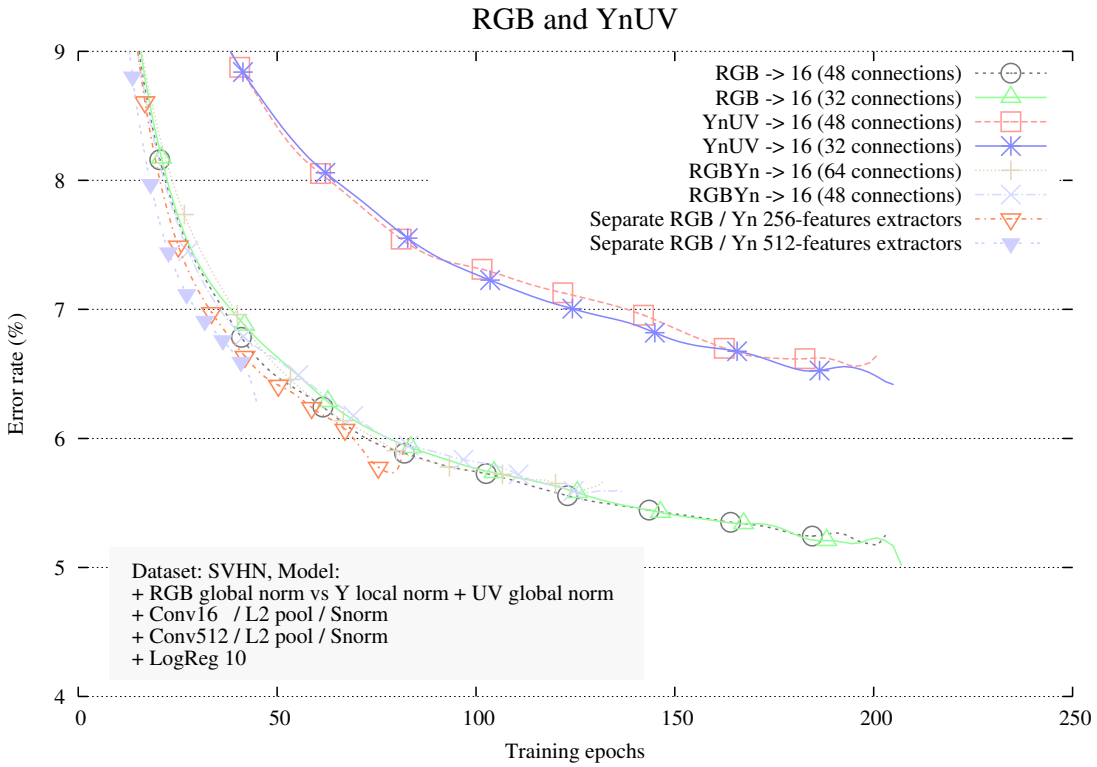

Figure 3.12: **Comparison of house numbers classification performance with (YnUV) and without preprocessing (RGB)**. Regardless of the connection scheme to the input channels, the non-preprocessed data (RGB) improves by 1 point over the preprocessed data.

Figure 3.13: **The benefits of local contrast normalization**: on the left, a sample taken from the GTSRB traffic sign dataset appears uniformly black to the human eye. Structure is actually revealed by local contrast normalization (bottom). Channels are displayed as follow: YnUV, Yn, U and V. The right side shows a normal sample's preprocessing.

3.3.4 Twisted tanh and Rectified Linear

A popular nonlinearity in neural networks is the $tanh()$ function. This sigmoid function however may be problematic when large inputs get stuck in the flat spots, where gradients of the function are very small. Hence moving along the sigmoid function to its opposite part while training may take a very long time. Adding a small linear term or *twisting term* as suggested by [9] may help avoiding this situation by making the flat spots steeper. In addition, similarly to the rectified linear units by [62], essentially a linear function for the positive inputs and zero for the negative inputs, the linear term can preserve a sense of relative intensity between units. The twisted tanh function is defined by:

$$f(x) = tanh(x) + \alpha x$$

where α is a small linear coefficient. We experiment with a range of values for α in Figure 3.15 and plot their corresponding shapes in Figure 3.14. In the context of the house numbers classification task, a twisted tanh function with $\alpha = 0.16$ consistently brought a non-negligible improvement over the regular $tanh$ sigmoid as shown in Figure 3.15. We also compare with using rectified linear non-linearity and find that it yields better results than the regular $tanh$ function, while giving similar results to the best twisted tanh configurations. Our experiment has a slight bias in that it also used momentum and multi-stage features for the rectified linear run only. Further work should establish a strict comparison on multiple datasets.

3.3.5 Momentum

Here we study the impact of momentum while training on the SVHN dataset. Figure 3.16 shows that high momentum is beneficial at the beginning of training but eventually has a negative impact while lower momentum values reach higher accuracies on the long term. While a carefully tuned momentum gradual decrease should be optimal, an intermediate momentum tuning such as 0.6 is a safe bet in the absence of tuned decrease.

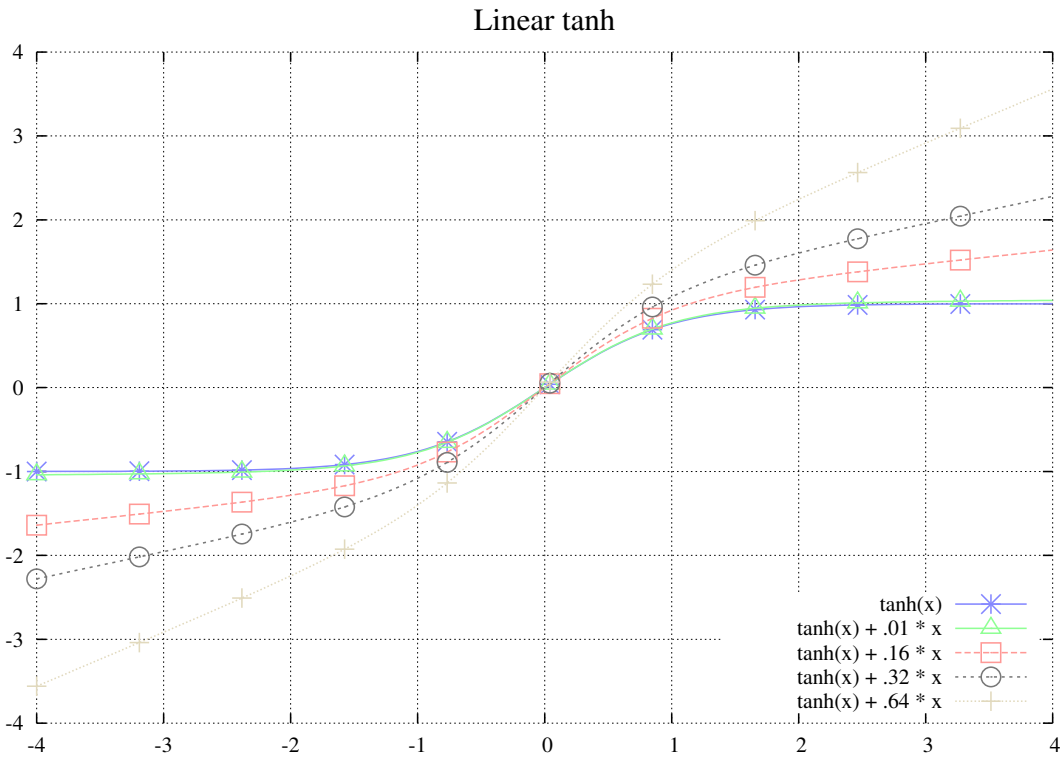

Figure 3.14: **Twisted tanh** shapes for different twisting term coefficients. A twisting coefficient of 0.16 is optimal on the SVHN dataset.

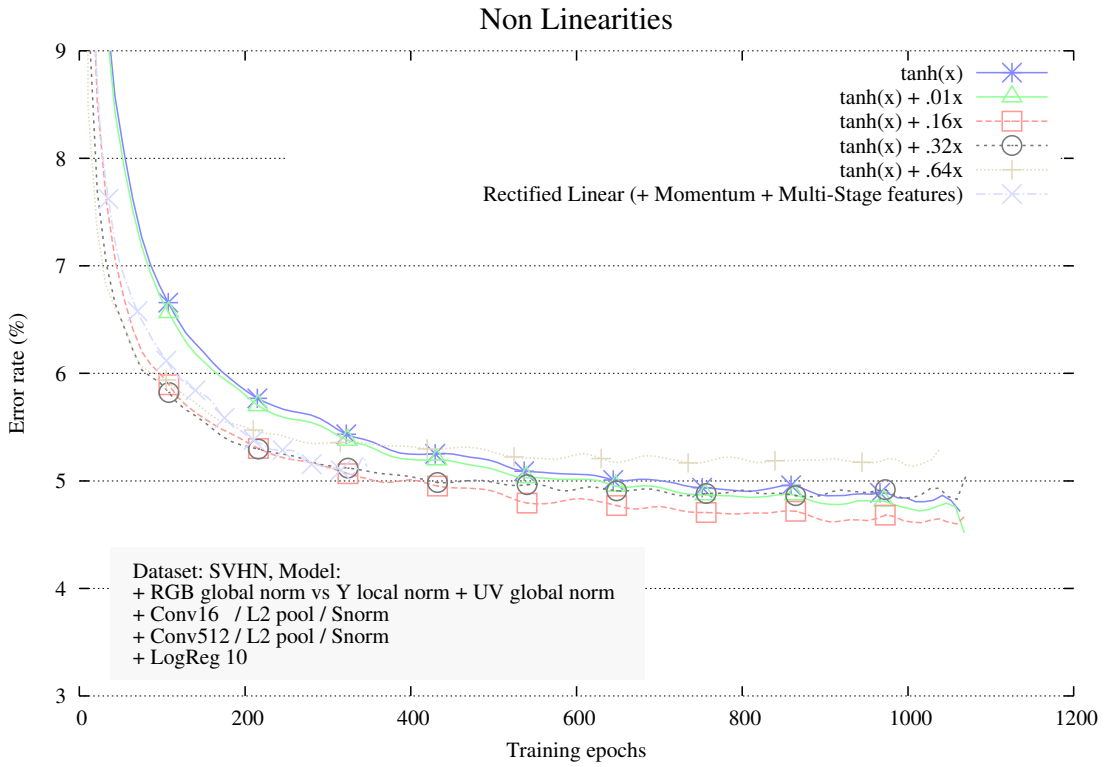

Figure 3.15: **The advantages of training with the twisted tanh nonlinearity** on the SVHN dataset over the regular $tanh()$ function. Twisted tanh with a linear coefficient of 0.16 brings a non-negligible improvement on validation accuracy.

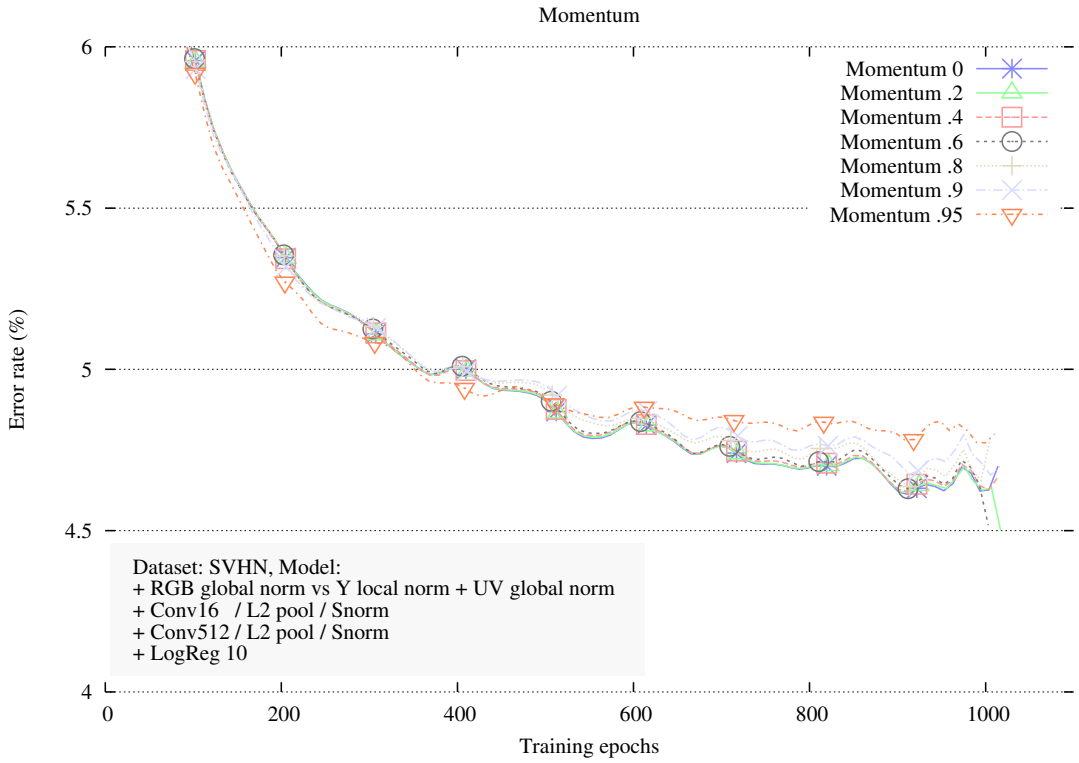

Figure 3.16: **The shape of training curves with different momentum values** reveals the early benefits of high momentum while lower momentum takes over as training continues.

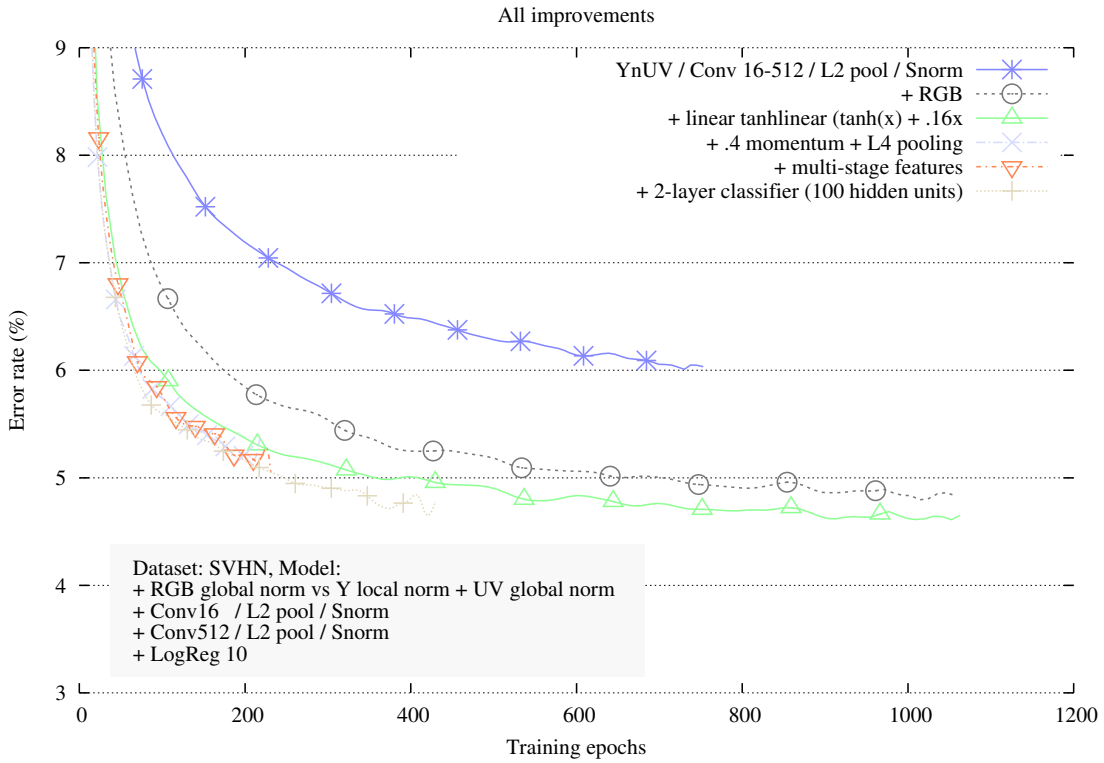

Figure 3.17: **A summary of architecture improvements over** [17]. From top to bottom we incrementally add architectural changes to the baseline model published in [17]. The biggest error rate decrease is induced by changing the normalized YUV input to an un-normalized RGB input, followed by the use of a twisted tanh over a regular tanh sigmoid nonlinearity. Minor improvements are then added with an intermediate momentum, L4 pooling instead of L2, multi-stage features and a 2-layer classifier.

3.4 Architecture Tuning

Different architectures can be optimal for different problems. Mainly, the more complex the problem, the more capacity and depth is necessary. However, too much capacity will lead to overfitting and the right balance must be found through experimentation. The main set of architectural hyper-parameters to explore usually comprises the number of stages, the number of features at each stage, the overall number of parameters or the shape of a network (e.g. increasing number of features or narrowing with a bottleneck). Eigen *et al.* [63] recently started to shed light on architecture tuning by using recursive neural networks to decouple the hyper-parameters, which we review below. These parameters have complex relationships, one can design models with the same total number of parameters by increasing the number of stages or the number of features at each stage, or use most of the parameters at the bottom or at the top of the hierarchy. We review here some important hyper-parameters:

1. **Number of stages and features.** In ConvNets, we call a stage a set of layers typically composed of a convolution layer, a non-linearity layer, an optional pooling layer and an optional normalization layer. These are followed by a classifier stage or a regression stage (multiple fully connected layers) for classification and regression problems. Traditionally, datasets with complexity similar to MNIST, GTSRB, SVHN or INRIA pedestrians (i.e. input sizes under 100 pixels in each dimension, limited number of viewpoints and less than 50 classes) were addressed with 2 feature extraction stages. However, [63, 64, 65] recently demonstrated accuracy gains by using deeper models. These gains can be partly explained by the rise of parallel processors, rendering the exploration of the hyper-parameters space less of a tedious exercise.

   Bigger problems such as ILSVRC13 (1000 classes, wide variety of viewpoints and backgrounds, much larger number of samples) have seen great successes using much

bigger and deeper models. For example, the model described in Table 3.1 uses 5 feature extraction stages and a total of approximately 50 million parameters, while the biggest model trained before the Krizhevsky model [21] used up to 10 million parameters at most [66, 20]. To address the 22,000 classes ImageNet dataset, [67] trained a 1 billion parameters network using a cluster of 16,000 cores. It should be noted that contrary to ConvNets, the filters were not shared across locations (i.e. not convolutional), hence inflating the number of parameters.

This recent model scaling was made possible by the advent of more powerful hardware (GPUs or large CPU clusters), bigger datasets (e.g. ImageNet) and better regularization techniques (e.g. DropOut [48] or MaxOut [64]). The ability to train models much faster has allowed a more systematic exploration of the hyper-parameters space and training of models with much bigger capacity than before. Additionally, large datasets and good regularization are crucial to prevent these large capacity models from overfitting. In the following section, we use an exhaustive search technique using a cpu cluster and subsequently improved results on the traffic sign dataset.

2. **Depth of classifier**. Multi-layer classifiers have historically used 1 or 2 layers in computer vision. This number has increased up to 3 or 4 layers in recent models for vision (Section 3.1) or speech (Section 3.2). Increasing the depth of the classifier allows natural formation of committees of independent sub-models when using Dropout ([48]). We show in Table 3.7 that Dropout can be employed on up to 4 layers and that it improves classification results.

3. **Network Shapes.** We call the shape of a ConvNet the outline drawn by its number of features by going up the hierarchy, but also refer to the filter sizes. [66] obtained very competitive results on MNIST with a pyramid-shaped Multi-Layer Perceptron (MLP), starting with a large base (2,500 units) and progressively

reducing to a small output (10 units). In speech recognition, [68] improve results by using a bottleneck shape with a Deep Neural Network (DNN), i.e. where the middle layer of a 5-layers network has a smaller number of features relative to the other layers. The bottleneck shape forces the network to learn useful information for classification in a low-dimensional space. This technique was initially devised in auto-encoders [69] for dimensionality reduction. In our experiments, we adopt the expanding shape with the intuition that low level filters only need to encode a small set of edges and colors, while features become more and more complex by representing larger and larger windows (from edges, to object parts, to parts, to scenes), thus requiring increasingly more features.

3.4.1 Exhaustive Random Model Selection

A number of important choices must be made regarding the architecture hyper-parameters as shown previously. A different approach to intuition from experts, is to search the hyper-parameters space exhaustively. However, since training end-to-end networks is expensive, this can only be achieved with inexpensive evaluations. [70] showed architecture choice is crucial in a number of state-of-the-art methods including ConvNets. They also demonstrate that the performance of randomly initialized architectures correlates with trained architecture performance when cross-validated. Using this idea, we can empirically search for an optimal architecture very quickly, by bypassing the time-consuming feature extractor training. We first extract features from a set of randomly initialized architectures with different capacities. We then train the top classifier using these features as inputs, again with a range of different capacities. In Figure 3.18, we train on the original (non jittered) GTSRB training set and evaluate against the validation set the following architecture parameters:

- Number of features at each stage: 108-108, 108-200, 38-64, 50-100, 72-128, 22-38 (the left and right numbers are the number of features at the first and second stages

respectively). Each convolution connection table has a density of approximately 70-80%, i.e. 108-8640, 108-16000, 38-1664, 50-4000, 72-7680, 22-684 in number of convolution kernels per stage respectively.

- Single or multi-scale features. The single-scale architecture (SS) uses only 2nd stage features as input to the classifier while the multi-stage architecture feed both the first and second stage outputs to the classifier (MS).
- Classifier architecture: single layer (fully connected) classifier or 2-layer (fully connected) classifier with the following number of hidden units: 10, 20, 50, 100, 200, 400.
- Color: we either use YUV channels or Y only.
- Different learning rates and regularization values.

The validation error curves in Figure 3.18 indicate that the most effective architecture is the multi-stage one without color information at a capacity around 1.5 million trainable parameters. A few of the best performing models with random weights are then fully trained and yielded an improvement over the first phase of the competition (which results are reported in Table 3.10, from 98.97% accuracy to 99.17% (see Table 3.11). This new record is established by increasing the classifier's capacity and depth (2-layer classifier with 100 hidden units instead of the single-layer classifier) and by ignoring color information (see corresponding convolutional filters in Fig 3.19).

We also evaluate the best ConvNet with random features in Table 3.11 (108-200 random features by training the 2-layer classifier with 100 hidden units only) and obtain 97.33% accuracy on the test set (see convolutional filters in Figure 3.19). Recall that this network was trained on the non-jittered dataset and could thus perform even better. The exact same architecture with trained features reaches 98.85% accuracy only while a network with a smaller second stage (108 instead of 200) reached 99.17%. Comparing

random and trained convolutional filters (Fig 3.19), we observe that 2^{nd} stage trained filters mostly contain flat surfaces with sparse positive or negative responses. While these filters are quite different from random filters, the 1^{st} stage trained filters are not. The specificity of the learned 2^{nd} stage filters may explain why more of them are required with random features, thus increasing the chances of containing appropriates features. A smaller 2^{nd} stage however may be easier to train with less diluted gradients and more optimal in terms of capacity. We therefore infer that after finding an optimal architecture with random features, one should try smaller stages (beyond the 1^{st} stage) with respect to the best random architecture, during full supervision.

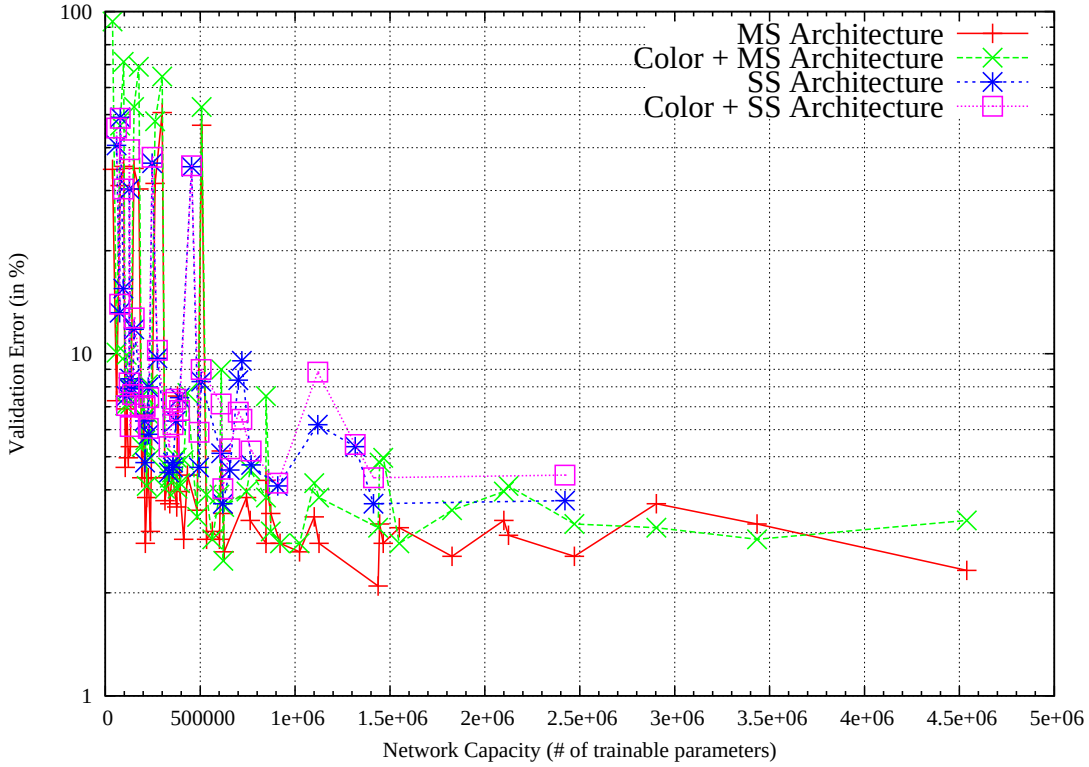

Figure 3.18: **Validation error rate of random-weights architectures** trained on the non-jittered dataset. The horizontal axis is the number of trainable parameters in the network. For readability, we group all architectures described in the text according to 2 variables: color and architecture (single or multi-stage).

Phase I #	Team	Method	Accuracy
197	IDSIA	cnn_hog3	98.98%
196	IDSIA	cnn_cnn_hog3	98.98%
178	**sermanet**	**EBLearn 2LConvNet ms 108 feats**	**98.97%**
195	IDSIA	cnn_cnn_hog3_haar	98.97%
187	**sermanet**	**EBLearn 2LConvNet ms 108 + val**	**98.89%**
199	INI-RTCV	Human performance	98.81%
170	IDSIA	ConvNet(IMG)_MLP(HOG3)	98.79%
177	IDSIA	ConvNet(IMG)_MLP(HOG3)_MLP(HAAR)	98.72%
26	**sermanet**	**EBLearn 2-layer ConvNet ms**	**98.59%**
193	IDSIA	ConvNet 7HL norm	98.46%
198	**sermanet**	**EBLearn 2-layer ConvNet ms reg**	**98.41%**
185	**sermanet**	**EBLearn 2L ConvNet ms + validation**	**98.41%**
27	**sermanet**	**EBLearn 2-layer ConvNet ss**	**98.20%**
191	IDSIA	ConvNet 7HL	98.10%
183	Radu.Timofte@VISICS	IKSVM+LDA+HOGs	97.88%
166	IDSIA	ConvNet 6HL	97.56%
184	Radu.Timofte@VISICS	CS+I+HOGs	97.35%

Table 3.10: **Top 17 test set accuracy results** during phase I of the GTSRB competition.

Phase I #	Team	Method	Accuracy
	sermanet	**EBLearn 2LConvNet ms 108-108 + 100-feats CF classifier + No color**	**99.17%**
197	IDSIA	cnn_hog3	98.98%
196	IDSIA	cnn_cnn_hog3	98.98%
178	**sermanet**	**EBLearn 2LConvNet ms 108-108**	**98.97%**
	sermanet	**EBLearn 2LConvNet ms 108-200 + 100-feats CF classifier + No color**	**98.85%**
	sermanet	**EBLearn 2LConvNet ms 108-200 + 100-feats CF classifier + No color + Ramdom features + No jitter**	**97.33%**

Table 3.11: **Post phase I networks** evaluated against the official test set break the previous 98.98% accuracy record with 99.17%.

As a side note, it is interesting that among a number of diverse vision systems, the top 13 ones during phase I of the GTSRB challenge all use ConvNets with at least 98.10% accuracy and that human performance (98.81%) is outperformed by 5 of these (Table 3.10).

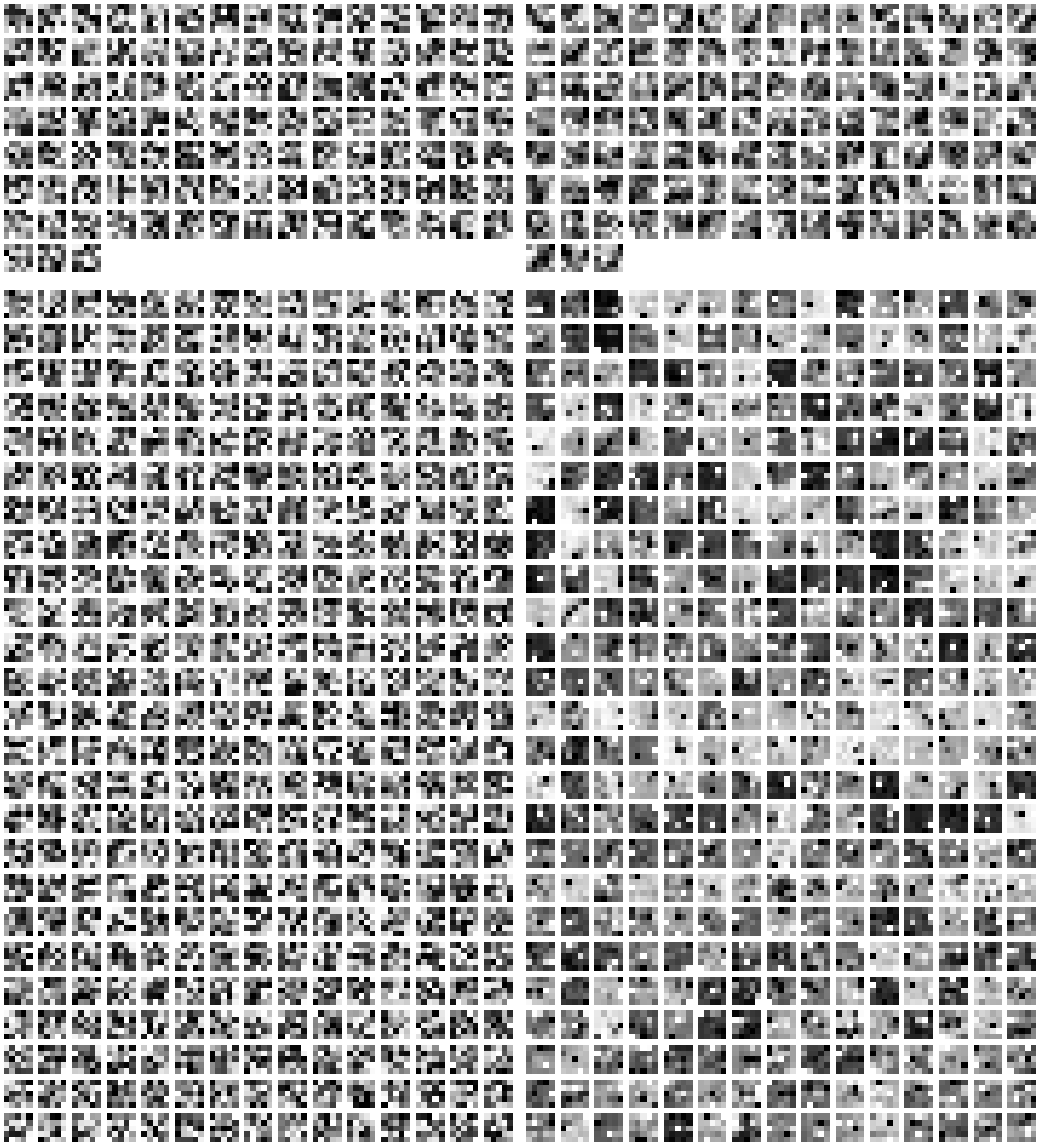

Figure 3.19: **5x5 convolution filters** for the first stage (top) and second stage (bottom). **Left:** Random-features ConvNet reaching 97.33% accuracy (see Table 3.11), with 108 and 16000 filters for stages 1 and 2 respectively. **Right:** Fully trained ConvNet reaching 99.17% accuracy, with 108 and 8640 filters for stages 1 and 2.

Chapter 4

Detection

4.1 Traditional Object Detection using ConvNets

In this section, we demonstrate a traditional approach to object detection using ConvNets. Traditional here refers to the bootstrapping passes and non-maximum suppression historically employed by many detection systems. In the following sections, we will depart slightly from these and devise a more effective approach.

Convolutional networks are efficient for detection because of their shared parameters, avoiding recomputation of features multiple times for different outputs. We demonstrate state of the art results on pedestrian detection datasets (INRIA [3] and Caltech [71]) and preliminary results on house numbers detection (SVHN [46]). Note that the pedestrian detection system described in subsequent sections was solely trained on INRIA data but also tested on the Caltech dataset, similarly to most other published systems. We also show the importance of multi-stage features and the combination of unsupervised and supervised techniques to obtain good performance.

4.1.1 Bootstrapping

Bootstrapping is typically used in detection settings by extracting the most offending negative answers and adding these samples multiple times to the existing dataset during training. For this purpose, we extract 3000 negative samples per bootstrapping pass on the INRIA dataset and limit the number of most offending answers to 5 for each image. We perform 3 bootstrapping passes in addition to the original training phase (i.e. totally 4 training passes).

4.1.2 Non-Maximum Suppression

Non-maximum suppression (NMS) is used to resolve conflicts when several bounding boxes overlap. For both INRIA and Caltech experiments we use the widely accepted PASCAL overlap criteria to determine a matching score between two bounding boxes ($\frac{intersection}{union}$) and if two boxes overlap by more than 60%, only the one with the highest score is kept. In [71]'s addendum, the matching criteria are modified by replacing the union of the two boxes with the minimum of the two. Therefore, if a box is fully contained in another one the small box is selected. The goal for this modification is to avoid false positives that are due to pedestrian body parts. However, a drawback to this approach is that it always disregards one of the overlapping pedestrians from detection. Instead of changing the criteria, we actively modify our training set before each bootstrapping phase. We include body part images that cause false positive detection into our bootstrapping image set. Our model can then learn to suppress such responses within a positive window and still detect pedestrians within bigger windows more reliably.

4.1.3 Color features

In the case of the Krizhevsky model, no special processing is used for color. However in previous work, we used a separate ConvNet channel in order to process the color channels more efficiently. Because color information in JPEG images (format used by most

pedestrian datasets) is coded at lower resolutions, we feed the color channels separately from the intensity channel. We convert all images into YUV image space and subsample the UV features. Then at the first stage, we keep feature extraction systems for Y and UV channels separate. On the Y channel, we use 32 7×7 features followed by absolute value rectification, contrast normalization and 3×3 subsampling. On the subsampled UV channels, we extract 6 5×5 features followed by absolute value rectification and contrast normalization, skipping the usual subsampling step since it was performed beforehand. These features are then concatenated to produce 38 feature maps that are input to the first layer. The second layer feature extraction takes 38 feature maps and produces 68 output features using 2040 9×9 features. A randomly selected 20% of the connections in mapping from input features to output features is removed to limit the computational requirements and break the symmetry [9]. The output of the second layer features are then transformed using absolute value rectification and contrast normalization followed by 2×2 subsampling. This results in a 17824 dimensional feature vector for each sample which is then fed into a linear classifier.

4.1.4 Pedestrian detection

The architecture used here is similar to the ones previously used for traffic sign and house numbers datasets, i.e. 2-stage ConvNets with multi-stage features. Additionally, we used unsupervised learning to initialize each stage, as described in the following section. We evaluate our system on all the major pedestrian detection benchmark datasets. We also show experiments that demonstrate the improvements coming from unsupervised training and multi-stage features. The ConvNet is trained on the INRIA pedestrian dataset [3]. Pedestrians are extracted into windows of 126 pixels in height and 78 pixels in width. The context ratio is 1.4, i.e. pedestrians are 90 pixels high and the remaining 36 pixels correspond to the background. Each pedestrian image is mirrored along the horizontal axis to expand the dataset. Similarly, we add 5 variations of each

original sample using 5 random deformations such as translations and scale. Translations range from -2 to 2 pixels and scale ratios from 0.95 to 1.05. These deformations enforce invariance to small deformations in the input. The range of each deformation determines the trade-off between recognition and localization accuracy during detection. An equal amount of background samples are extracted at random from the negative images and taking approximately 10% of the extracted samples for validation yields a validation set with 2000 samples and training set with 21845 samples.

4.1.4.1 Unsupervised Learning using Convolutional Sparse Coding

Recently sparse coding has seen much interest in many fields due to its ability to extract useful feature representations from data, The general formulation of sparse coding is a linear reconstruction model using an over-complete dictionary $\mathcal{D} \in \mathcal{R}^{m \times n}$ where $m > n$ and a regularization penalty on the mixing coefficients $z \in \mathcal{R}^n$.

$$z^* = \arg\min_z \|x - \mathcal{D}z\|_2^2 + \lambda s(z) \tag{4.1}$$

The aim is to minimize equation 4.1 with respect to z to obtain the optimal sparse representation z^* that correspond to input $x \in \mathcal{R}^m$. The exact form of $s(z)$ depends on the particular sparse coding algorithm that is used, here, we use the $\|.\|_1$ norm penalty, which is the sum of the absolute values of all elements of z. It is immediately clear that the solution of this system requires an optimization process. Many efficient algorithms for solving the above convex system has been proposed in recent years [72, 73, 74, 75]. However, our aim is to also learn generic feature extractors. For that reason we minimize equation 4.1 wrt $\mathcal{D}$ too.

$$z^*, \mathcal{D}^* = \arg\min_{z,\mathcal{D}} \|x - \mathcal{D}z\|_2^2 + \lambda \|z\|_1 \tag{4.2}$$

This resulting equation is non-convex in $\mathcal{D}$ and z at the same time, however keeping one fixed, the problem is still convex wrt to the other variable. All sparse modeling algorithms that adopt the dictionary matrix $\mathcal{D}$ exploit this property and perform a coordinate descent like minimization process where each variable is updated in succession. Following [76] many authors have used sparse dictionary learning to represent images [77, 72, 78]. However, most of the sparse coding models use small image patches as input x to learn the dictionary $\mathcal{D}$ and then apply the resulting model to every overlapping patch location on the full image. This approach assumes that the sparse representation for two neighboring patches with a single pixel shift is completely independent, thus produces very redundant representations. [14, 79] have introduced convolutional sparse modeling formulations for feature learning and object recognition and we use the Convolutional Predictive Sparse Decomposition (CPSD) model proposed in [14] since it is the only convolutional sparse coding model providing a fast predictor function that is suitable for building multi-stage feature representations. The particular predictor function we use is similar to a single layer ConvNet of the following form:

$$f(x; g, k, b) = \tilde{z} = \{\tilde{z}_j\}_{j=1..n} \tag{4.3}$$

$$\tilde{z}_j = g_j \times \tanh(x \otimes k_j + b_j) \tag{4.4}$$

where $\otimes$ operator represents convolution operator that applies on a single input and single filter. In this formulation x is a $p \times p$ grayscale input image, $k \in \mathcal{R}^{n \times m \times m}$ is a set of 2D filters where each filter is $k_j \in \mathcal{R}^{m \times m}$, $g \in \mathcal{R}^n$ and $b \in \mathcal{R}^n$ are vectors with n elements, the predictor output $\tilde{z} \in \mathcal{R}^{n \times p-m+1 \times p-m+1}$ is a set of feature maps where each of $\tilde{z}_j$ is of size $p - m + 1 \times p - m + 1$. Considering this general predictor function, the final form of the convolutional unsupervised energy for grayscale inputs is as follows:

$$\mathbb{E}_{CPSD} = \mathbb{E}_{ConvSC} + \beta \mathbb{E}_{Pred} \tag{4.5}$$

$$\mathbb{E}_{ConvSC} = \left\| x - \sum\nolimits_j \mathcal{D}_j \otimes z_j \right\|_2^2 + \lambda \|z\|_1 \tag{4.6}$$

$$\mathbb{E}_{Pred} = \|z^* - f(x; g, k, b)\|_2^2 \tag{4.7}$$

where $\mathcal{D}$ is a dictionary of filters the same size as k and β is a hyper-parameter. The unsupervised learning procedure is a two step coordinate descent process. At each iteration, **(1) Inference:** The parameters $W = \{\mathcal{D}, g, k, b\}$ are kept fixed and equation 4.6 is minimized to obtain the optimal sparse representation z^*, **(2) Update:** Keeping z^* fixed, the parameters W updated using a stochastic gradient step: $W \leftarrow W - \eta \frac{\partial \mathbb{E}_{CPSD}}{\partial W}$ where η is the learning rate parameter. The inference procedure requires us to carry out the sparse coding problem solution. For this step we use the FISTA method proposed in [74]. This method is an extension of the original iterative shrinkage and thresholding algorithm [73] using an improved step size calculation with a momentum-like term. We apply the FISTA algorithm in the image domain adopting the convolutional formulation.

For color images or other multi-modal feature representations, the input x is a set of feature maps indexed by i and the representation z is a set of feature maps indexed by j for each input map i. We define a map of connections P from input x to features z. A j^{th} output feature map is connected to a set P_j of input feature maps. Thus, the predictor function in Algorithm 1 is defined as:

$$\tilde{z}_j = g_j \times \tanh\left(\sum_{i \in P_j} (x_i \otimes k_{j,i}) + b_j \right) \tag{4.8}$$

and the reconstruction is computed using the inverse map $\bar{P}$:

$$\mathbb{E}_{ConvSC} = \sum_i \|x_i - \sum_{j \in \bar{P}_i} \mathcal{D}_{i,j} \otimes z_j\|_2^2 + \lambda \|z\|_1 \tag{4.9}$$

For a fully connected layer, all the input features are connected to all the output features, however it is also common to use sparse connection maps to reduce the number of parameters. The online training algorithm for unsupervised training of a single layer is:

Algorithm 1 Single layer unsupervised training.

function Unsup$(x, \mathcal{D}, P, \{\lambda, \beta\}, \{g, k, b\}, \eta)$
  Set: $f(x; g, k, b)$ from eqn 4.8, $W^p = \{g, k, b\}$.
  Initialize: $z = \varnothing$, $\mathcal{D}$ and W^p randomly.
  repeat
    Perform **inference**, minimize equation 4.9 wrt z using FISTA [74]
    Do a stochastic **update** on $\mathcal{D}$ and W^p. $\mathcal{D} \leftarrow \mathcal{D} - \eta \frac{\partial \mathbb{E}_{ConvSC}}{\partial \mathcal{D}}$ and $W^p \leftarrow W^p - \eta \frac{\partial \mathbb{E}_{Pred}}{\partial W^p}$
  until convergence
  Return: $\{\mathcal{D}, g, k, b\}$
end function

4.1.4.2 Non-Linear Transformations

Once the unsupervised learning for a single stage is completed, the next stage is trained on the feature representation from the previous one. In order to obtain the feature representation for the next stage, we use the predictor function $f(x)$ followed by non-linear transformations and pooling. Following the multi-stage framework used in [14], we apply absolute value rectification, local contrast normalization and average down-sampling operations.

Absolute Value Rectification is applied component-wise to the whole feature output from $f(x)$ in order to avoid cancellation problems in contrast normalization and pooling steps.

Local Contrast Normalization is a non-linear process that enhances the most active

feature and suppresses the other ones. The exact form of the operation is as follows:

$$v_i = x_i - x_i \otimes w \ , \quad \sigma = \sqrt{\sum_i w \otimes v_i^2} \tag{4.10}$$

$$y_i = \frac{v_i}{\max(c, \sigma)} \tag{4.11}$$

where i is the feature map index and w is a 9×9 Gaussian weighting function with normalized weights so that $\sum_{ipq} w_{pq} = 1$. For each sample, the constant c is set to $mean(\sigma)$ in the experiments.

Average Down-Sampling operation is performed using a fixed size boxcar kernel with a certain step size. The size of the kernel and the stride are given for each experiment in the following sections.

Once a single layer of the network is trained, the features for training a successive layer is extracted using the predictor function followed by non-linear transformations. Detailed procedure of training an N layer hierarchical model is explained in Algorithm 2.

Algorithm 2 Multi-layer unsupervised training.

function HierarUnsup$(x, n_i, m_i, P_i, \{\lambda_i, \beta_i\}, \{w_i, s_i\},$
$i = \{1..N\}, \eta_i)$
  Set: $i = 1$, $X_1 = x$, $lcn(x)$ using equations 4.10-4.11, $ds(X, w, s)$ as the down-sampling operator using boxcar kernel of size $w \times w$ and stride of size s in both directions.
  repeat
    Set: $\mathcal{D}_i, k_i \in \mathcal{R}^{n_i \times m_i \times m_i}$, $g_i, b_i \in \mathcal{R}^{n_i}$.
    $\{\mathcal{D}_i, k_i, g_i, k_i, b_i\} =$
      $Unsup(X_i, \mathcal{D}_i, P_i, \{\lambda_i, \beta_i\}, \{g_i, k_i, b_i\}, \eta_i)$
    $\tilde{z} = f(X_i; g_i, k_i, b_i)$ using equation 4.8.
    $\tilde{z} = |\tilde{z}|$
    $\tilde{z} = lcn(\tilde{z})$
    $X_{i+1} = ds(\tilde{z}, w_i, s_i)$
    $i = i + 1$
  until $i = N$
end function

The first layer features can be easily displayed in the parameter space since the

parameter space and the input space is same, however visualizing the second and higher level features in the input space can only be possible when only invertible operations are used in between layers. However, since we use absolute value rectification and local contrast normalization operations mapping the second layer features onto input space is not possible. In Figure 4.2 we show a subset of 1664 second layer features in the parameter space.

4.1.4.3 Evaluation Protocol

During testing and bootstrapping phases using the INRIA dataset, the images are both up-sampled and sub-sampled. The up-sampling ratio is 1.3 while the sub-sampling ratio is limited by 0.75 times the network's minimum input (126×78). We use a scale stride of 1.10 between each scale, while other methods typically use either 1.05 or 1.20 [71]. A higher scale stride is desirable as it implies less computations.

For evaluation we use the bounding boxes files published on the Caltech Pedestrian website [1] and the evaluation software provided by Piotr Dollar (version 3.0.1). In an effort to provide a more accurate evaluation, we improved on both the evaluation formula and the INRIA annotations as follows. The evaluation software was slightly modified to compute the full area under curve (AUC) in the entire [0, 1] range rather than from 9 discrete points only (0.01, 0.0178, 0.0316, 0.0562, 0.1, 0.1778, 0.3162, 0.5623 and 1.0 in version 3.0.1). Instead, we compute the entire area under the curve by summing the areas under the piece-wise linear interpolation of the curve, between each pair of points. In addition, we also report a 'fixed' version of the annotations for INRIA dataset, which has missing positive labels. The added labels are only used to avoid counting false errors and wrongly penalizing algorithms. The modified code and extra INRIA labels are available at [2] [A. Table 4.1 reports results for both original and fixed INRIA datasets. Notice that the full AUC and fixed INRIA annotations both yield a reordering of the results.

[1] http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians
[2] http://cs.nyu.edu/~sermanet/data.html#inria

To ensure a fair comparison, we separated systems trained on INRIA (the majority) from systems trained on TUD-MotionPairs and the only system trained on Caltech in table 4.1. For clarity, only systems trained on INRIA were represented in Figure 4.3 and Figure 4.4, however all results for all systems are still reported in table 4.1.

4.1.4.4 Results

In Figure 4.1, we plot DET curves, i.e. miss rate versus false positives per image (FPPI), on the fixed INRIA dataset and rank algorithms along two measures: the error rate at 1 FPPI and the area under curve (AUC) rate in the [0, 1] FPPI range. For both measures, lower is better. This graph shows the individual contributions of unsupervised learning (ConvNet-U) and multi-stage features learning (ConvNet-F-MS) and their combination (ConvNet-U-MS) compared to the fully-supervised system without multi-stage features (ConvNet-F). Considering the AUC measure, unsupervised learning exhibits the most improvements with 17.81% error compared to the baseline ConvNet (26.05%). Multi-stage features without unsupervised learning reach 20.43% error while their combination yields the state of the art error rate of 11.05%.

An extensive comparison of results for all major pedestrian datasets and published systems is provided in Table 4.1. Multiple types of measures proposed by [71] are reported. For clarity, we also plot in Figure 4.3 and Figure 4.4 two of these measures, 'reasonable' and 'large', for INRIA-trained systems. The 'large' plot shows that the ConvNet results in state-of-the-art performance with some margin on the ETH, Caltech and TudBrussels datasets and is closely behind LatSvm-V2 and VeryFast for INRIA and Daimler datasets. In the 'reasonable' plot, the ConvNet yields competitive results for INRIA, Daimler and ETH datasets but performs poorly on the Caltech dataset. We suspect the ConvNet with multi-stage features trained at high-resolution is more sensitive to resolution loss than other methods. In future work, a ConvNet trained at multiple resolutions will likely learn to use appropriate cues for each resolution regime.

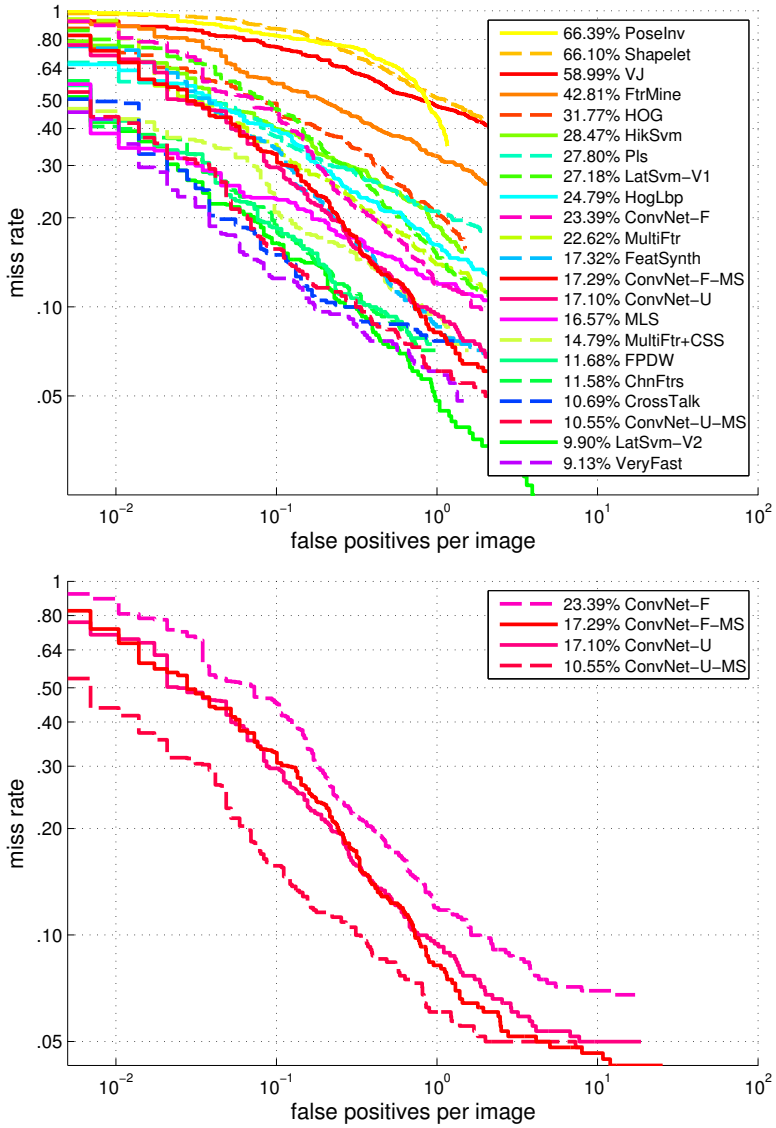

Figure 4.1: **DET curves on the fixed-INRIA dataset for large pedestrians measure** report false positives per image (FPPI) against miss rate. Algorithms are sorted from top to bottom using the proposed continuous area under curve measure between 0 and 1 FPPI. **Below, only the ConvNet variants are displayed to highlight the individual contributions of unsupervised learning (ConvNet-U) and multi-stage features learning (ConvNet-F-MS) and their combination (ConvNet-U-MS) compared to the fully-supervised system without multi-stage features (ConvNet-F).**

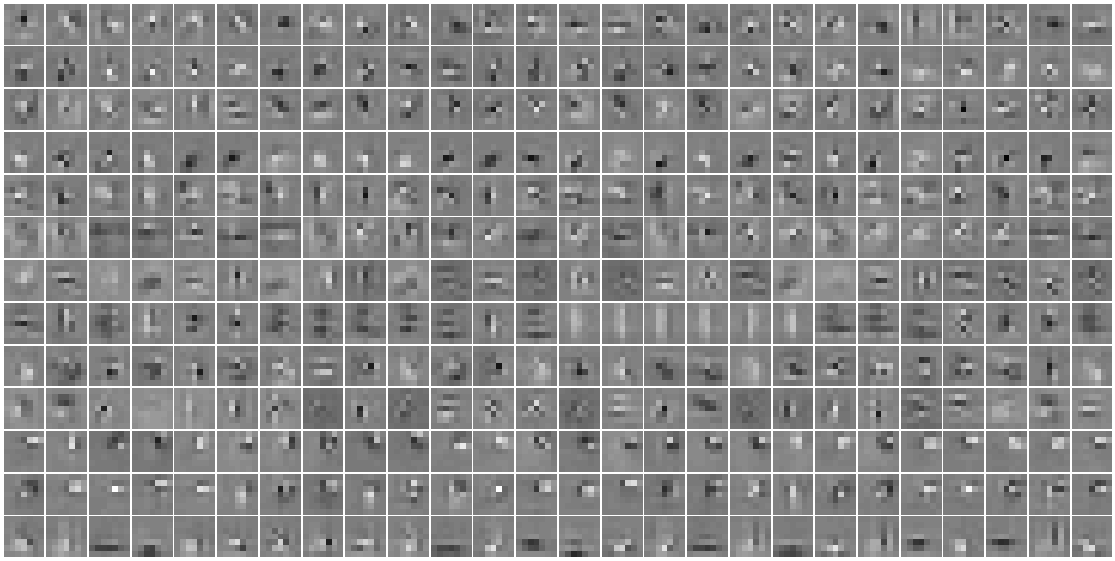

Figure 4.2: **A subset of 7×7 second layer filters** trained on grayscale INRIA images using Algorithm 2. Each row in the figure shows filters that connect to a common output feature map. It can be seen that they extract features at similar locations and shapes, e.g. the bottom row tends to aggregate horizontal features towards the bottom of the filters.

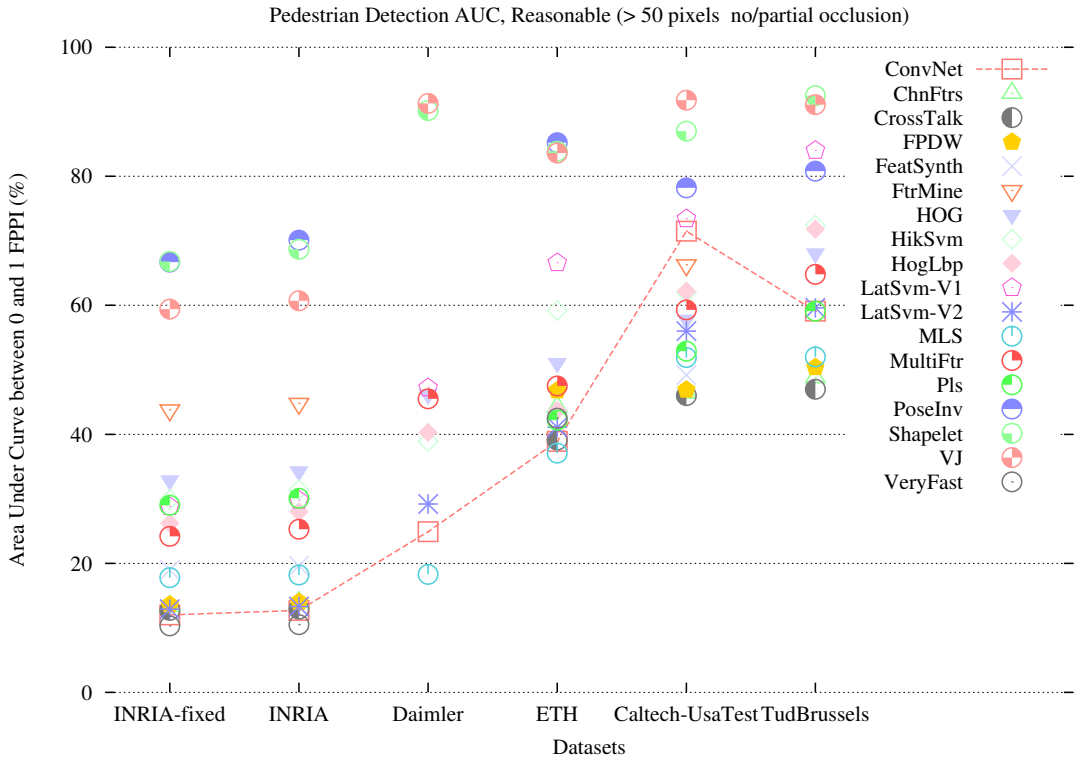

Figure 4.3: AUC percentage of all INRIA-trained systems on all major datasets (INRIA-fixed, INRIA, Daimler, ETH, Caltech-UsaTest and TudBrussels) using the *reasonable* measure. The AUC is computed from DET curves (smaller AUC means more accuracy and less false positives). For clarity, each ConvNet performance is connected by dotted lines. Only the 'reasonable' measure is plotted here, however all measures are reported in table 4.1. For this measure, the ConvNet system yields state-of-the-art or competitive results on most datasets, except for the Caltech dataset which contains many low-resolution images. We hypothesize that our ConvNet relies more on high-resolution cues than other methods. This is hinted in Figure 4.4 by the state of the art result obtained on the same Caltech datasets using the *large* measure only (i.e. high-resolution pedestrians).

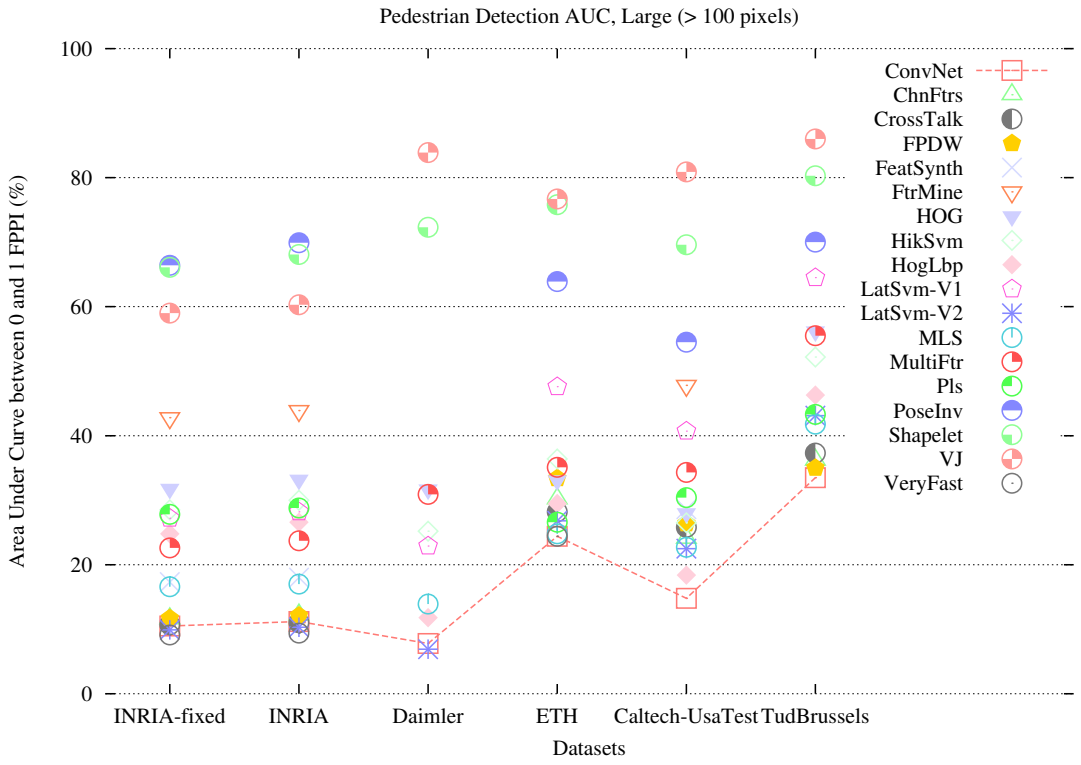

Figure 4.4: AUC percentage of all INRIA-trained systems on all major datasets using the *large* measure. Contrary to Figure 4.3, our ConvNet performs very competitively on all datasets. This suggests that this ConvNet is particularly good in the high-resolution regime.

Trained on	INRIA																		TUD-MotionPairs		Caltech
	Conv Net	**Chn Ftrs**	**Cross Talk**	**FPDW**	**Feat Synth**	**Ftr Mine**	**HOG**	**Hik Svm**	**Hog Lbp**	**LatSvm -V1**	**LatSvm -V2**	**MLS**	**Multi Ftr**	**Pls**	**Pose Inv**	**Shapelet**	**VJ**	**Very Fast**	**MultiFtr +CSS**	**MultiFtr +Motion**	**Multi ResC**
All - AUC %																					
INRIA-fixed	12.0	13.3	12.7	13.6	19.0	43.8	32.9	29.9	26.2	28.8	12.9	17.8	24.2	29.0	66.7	66.8	59.4	**10.3**	16.1	-	-
INRIA	12.7	13.9	12.9	14.0	19.6	44.8	34.3	31.4	28.0	29.8	13.3	18.2	25.3	30.1	70.1	68.7	60.7	**10.5**	16.9	-	-
Daimler	58.6	-	-	-	-	-	67.9	62.4	69.8	64.2	62.3	**51.8**	68.8	-	-	94.9	94.8	-	48.6	40.5	-
ETH	47.1	48.7	43.8	51.5	-	-	54.9	61.6	51.1	69.1	49.3	**42.8**	51.7	47.4	86.5	85.6	84.5	46.9	59.5	58.2	-
Caltech-UsaTest	90.9	**77.1**	77.8	78.1	78.1	86.7	85.5	86.8	87.9	91.7	84.2	83.4	83.4	81.2	92.6	95.4	99.1	-	81.2	77.9	74.2
TudBrussels	66.8	57.6	**55.0**	59.0	-	-	73.6	76.4	77.2	85.7	67.2	59.2	70.5	66.1	83.8	93.8	92.7	-	57.8	51.4	-
Reasonable - AUC % - >50 pixels & no/partial occlusion																					
INRIA-fixed	12.0	13.3	12.7	13.6	19.0	43.8	32.9	29.9	26.2	28.8	12.9	17.8	24.2	29.0	66.7	66.8	59.4	**10.3**	16.1	-	-
INRIA	12.7	13.9	12.9	14.0	19.6	44.8	34.3	31.4	28.0	29.8	13.3	18.2	25.3	30.1	70.1	68.7	60.7	**10.5**	16.9	-	-
Daimler	24.9	-	-	-	-	-	46.2	38.9	40.3	47.2	29.2	**18.3**	45.5	-	-	90.2	91.3	-	29.0	23.3	-
ETH	38.9	44.2	39.1	46.8	-	-	51.1	59.2	43.7	66.6	41.1	**37.1**	47.5	42.2	85.2	83.9	83.6	42.5	49.4	47.9	-
Caltech-UsaTest	71.5	46.4	**46.0**	46.9	49.2	66.3	57.8	62.0	62.2	73.4	56.0	51.9	59.3	52.9	78.2	87.0	91.8	-	52.1	42.3	38.1
TudBrussels	59.1	48.8	**47.0**	50.4	-	-	68.1	72.4	71.8	84.0	59.6	52.0	64.8	59.1	80.8	92.5	91.1	-	50.2	43.8	-
Large - AUC % - >100 pixels																					
INRIA-fixed	10.5	11.6	10.7	11.7	17.3	42.8	31.8	28.5	24.8	27.2	9.9	16.6	22.6	27.8	66.4	66.1	59.0	**9.1**	14.8	-	-
INRIA	11.2	12.2	11.0	12.1	18.0	43.9	33.2	30.0	26.6	28.2	10.3	17.0	23.7	28.8	69.9	68.1	60.3	**9.4**	15.6	-	-
Daimler	7.8	-	-	-	-	-	31.7	25.2	11.8	22.9	**6.9**	13.9	30.9	-	-	72.3	83.9	-	18.7	18.6	-
ETH	**24.4**	30.2	28.2	33.4	-	-	33.1	36.4	29.5	47.6	26.8	24.8	35.1	26.6	63.9	75.8	76.7	24.4	36.7	31.6	-
Caltech-UsaTest	**14.8**	24.1	25.8	26.4	28.6	47.8	28.0	26.5	18.4	40.7	22.5	22.7	34.3	30.4	54.5	69.6	80.9	-	28.8	12.0	12.5
TudBrussels	**33.5**	36.2	37.3	35.0	-	-	56.2	52.2	46.3	64.5	43.1	41.8	55.5	43.3	70.0	80.3	86.0	-	45.3	38.1	-
Near - AUC % - >80 pixels																					
INRIA-fixed	11.3	11.6	11.0	11.9	17.3	42.6	31.5	28.5	24.7	27.5	11.1	16.5	22.7	27.7	66.0	66.1	58.7	**9.7**	14.7	-	-
INRIA	11.9	12.2	11.2	12.3	17.9	43.7	32.9	30.0	26.5	28.5	11.5	16.8	23.8	28.8	69.4	68.1	60.0	**9.9**	15.5	-	-
Daimler	**10.0**	-	-	-	-	-	36.8	30.4	10.9	27.6	10.8	14.7	33.7	-	-	78.3	86.3	-	18.4	19.5	-
ETH	**28.9**	35.2	30.9	37.5	-	-	40.5	45.6	31.7	52.2	31.4	29.5	39.4	34.1	80.6	79.9	80.0	29.8	40.0	36.3	-
Caltech-UsaTest	27.3	27.4	28.9	28.4	29.5	48.9	33.1	34.3	**24.7**	47.2	26.7	29.1	40.8	31.2	66.8	75.7	85.3	-	30.4	16.4	15.0
TudBrussels	40.4	39.5	40.3	**38.8**	-	-	61.1	58.7	50.5	70.9	47.1	45.3	57.2	49.6	80.0	85.6	89.0	-	46.5	39.8	-
Medium - AUC % - 30-80 pixels																					
INRIA-fixed	33.1	100.0	99.7	100.0	100.0	100.0	100.0	100.0	85.3	85.3	99.7	100.0	86.1	100.0	99.7	99.7	91.5	**27.9**	91.3	-	-
INRIA	33.1	100.0	99.7	100.0	100.0	100.0	100.0	100.0	85.3	85.3	99.7	100.0	86.1	100.0	99.7	99.7	91.5	**27.9**	91.3	-	-
Daimler	54.2	-	-	-	-	-	62.1	54.4	70.7	58.5	60.0	**44.7**	63.2	-	-	95.2	93.7	-	43.4	34.0	-
ETH	55.4	42.9	**42.1**	45.4	-	-	49.9	54.7	61.2	71.5	57.3	43.9	47.3	45.0	73.9	74.5	71.2	48.3	55.2	54.7	-
Caltech-UsaTest	92.2	**69.5**	70.6	70.6	70.2	82.1	81.4	82.6	91.5	91.1	80.8	80.6	77.8	75.8	88.8	94.7	98.7	-	76.0	73.4	65.4
TudBrussels	67.8	57.4	**55.5**	59.7	-	-	71.4	74.9	82.9	85.5	68.2	59.1	68.7	65.0	79.4	94.1	91.7	-	55.0	48.6	-
Far - AUC % - 20-30 pixels																					
Caltech-UsaTest	100.0	**93.4**	95.4	94.2	94.7	95.0	96.2	98.2	100.0	97.8	97.4	100.0	95.9	98.7	100.0	99.9	99.3	-	95.3	94.6	100.0
Partial occlusion - AUC % - >50 pixels																					
Caltech-UsaTest	81.9	**61.3**	69.2	66.9	67.0	81.6	77.1	80.3	75.0	84.1	76.7	71.3	78.6	68.1	85.7	90.5	96.9	-	76.6	64.4	61.3
Heavy occlusion - AUC % - >50 pixels																					
Caltech-UsaTest	96.3	91.3	90.7	91.8	**87.8**	95.5	93.7	93.2	95.6	94.4	93.3	92.1	95.0	92.3	97.2	97.3	98.2	-	91.1	87.8	84.0
Atypical aspect ratio - AUC % - >50 pixels, no occlusion																					
Caltech-UsaTest	74.5	57.3	**55.4**	61.9	60.3	76.8	75.3	77.4	77.2	85.8	70.8	71.5	73.3	69.1	86.0	92.1	93.6	-	64.3	48.9	50.6

Table 4.1: This table reports the performance of all systems on all datasets using the full AUC percentage over the considered range [0,1] from DET curves. DET curves plot false positives per image (FPPI) against miss rate. Hence a smaller AUC% means a more accurate system with greater reduction of false positives. Top performing results (INRIA-trained only) are highlighted in bold. We report the multiple measures introduced by [71] for all major pedestrian datasets. The far, occlusion and aspect-ratio measures are only available for the Caltech dataset.

Figure 4.5: **Detection results on a patchwork of worst answers from SVHN classification**. Spurious detections of "1" are present but these can be explained (Figure 4.6). However many digits are correctly detected while these were the most offending answers by the same system run in classification mode. These correct classifications when alleviating the scaling issue via detection can probably explain most of the remaining errors of the system.

4.1.5 House Numbers Detection

This preliminary work reuses successful techniques on pedestrian detection and house numbers classification to perform house numbers detection, the eventual practical application targeted by the SVHN dataset.

The pure classification model trained earlier on SVHN was reused in a detection setting even though it was not trained with a background class nor underwent bootstrapping passes and yet demonstrates promising results. Figure 4.5 is a patchwork of the worst errors (errors with highest confidence) from this classification model. This patchwork is a single image passed through our detector and hence is not a realistic image, rather a simple experiment. The artificial grid structure resulting from this patchwork confuses the network into detecting many instances of class "1" because of its vertical structure. This explains the extraneous "1" detections and can be verified by the spurious activations in the detection outputs maps in Figure 4.6 (second column of output maps from the left). However, an important number of digits are correctly detected which is

promising for a model fully trained for a detection setting. It also indicates that the remaining errors in the classification setting and slightly lower performance compared to humans are not due to shortcomings of the model but rather to large scale variations that humans naturally cope with by performing detection. Thus our detection system run on the classification task or directly on the detection task will likely exceed human performance as it did before for traffic sign classification (Table 3.10).

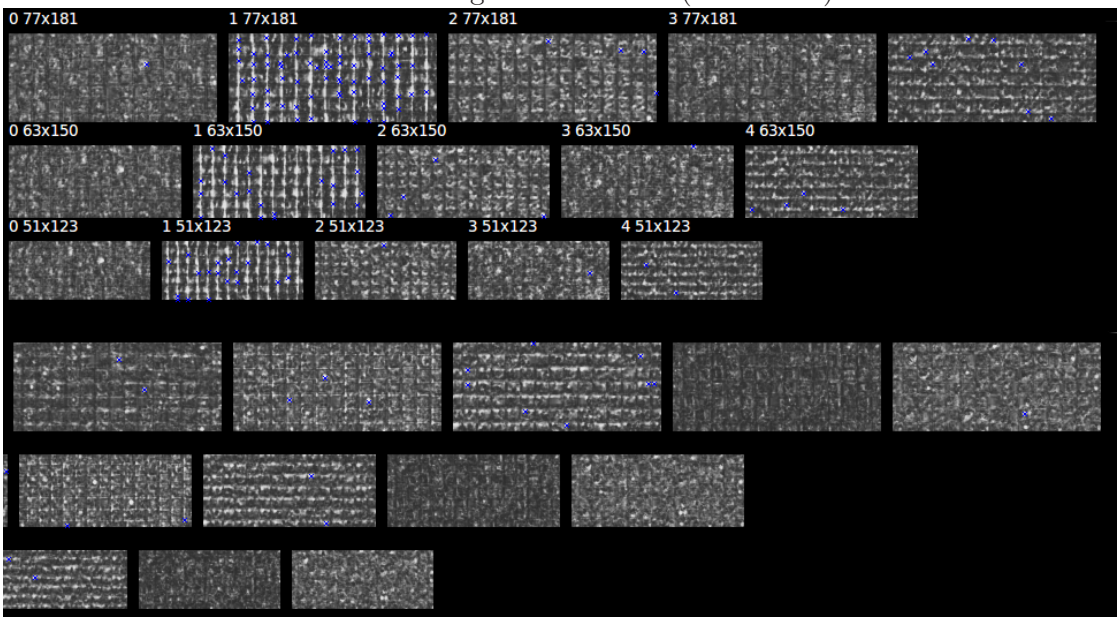

Figure 4.6: **Detection outputs maps on a patchwork of worst answers from SVHN classifications**. These outputs maps represent the activation maps for each class, black meaning no activation and white meaning full activation. The maps are organized by class, the first column standing for class "0", the second for class "1", etc. The rows result from different input scales, the top row being the high resolution scale. The artificial grid structure of the patchwork image can be seen here for class "1" where lots of white activations are present and can be explained by the resemblance between the vertical grid and the shape of digit "1". These spurious activations would not be present on natural images and differentiation between class "1" and vertical edges would be trained for during bootstrapping.

4.2 ConvNets and Sliding Window Efficiency

ConvNets are efficient in terms of learning because sharing the weights at multiple locations regularizes the filters to be more general and speeds up learning by accumulating more gradients. But by nature, ConvNets are also computationally efficient when applied densely, i.e. no redundant computations are performed, as opposed to other architectures that have to recompute the entire pipeline for each output unit. For ConvNets, neighboring output units share common inputs in lower layers. For example, applying a ConvNet to its minimum window size will produce a spatial output size of 1x1, as in Figure 4.7. Extending to outputs of size 2x2 requires recomputation of only a minimal part of the features (yellow region in Figure 4.7).

Note that while the last layers of our architecture are fully connected linear layers, during detection these layers are effectively replaced by convolution operations with kernels of 1x1. Then the entire ConvNet is simply a sequence of convolutions, max-pooling and thresholding operations only.

4.3 Object Localization

Starting from our classification-trained network, we replace the classifier layers by a regression network and train it to predict object bounding boxes at each spatial location and scale. We then combine the regression predictions together into objects and in turn combine these with the classification results of each location, as we now describe.

4.3.1 Generating Predictions

To generate object bounding box predictions, we simultaneously run the classifier and regressor networks across all locations and scales. Since these share the same feature extraction layers, only the final regression layers need to be recomputed after computing the classification network. The output of the final softmax layer for a class c at each

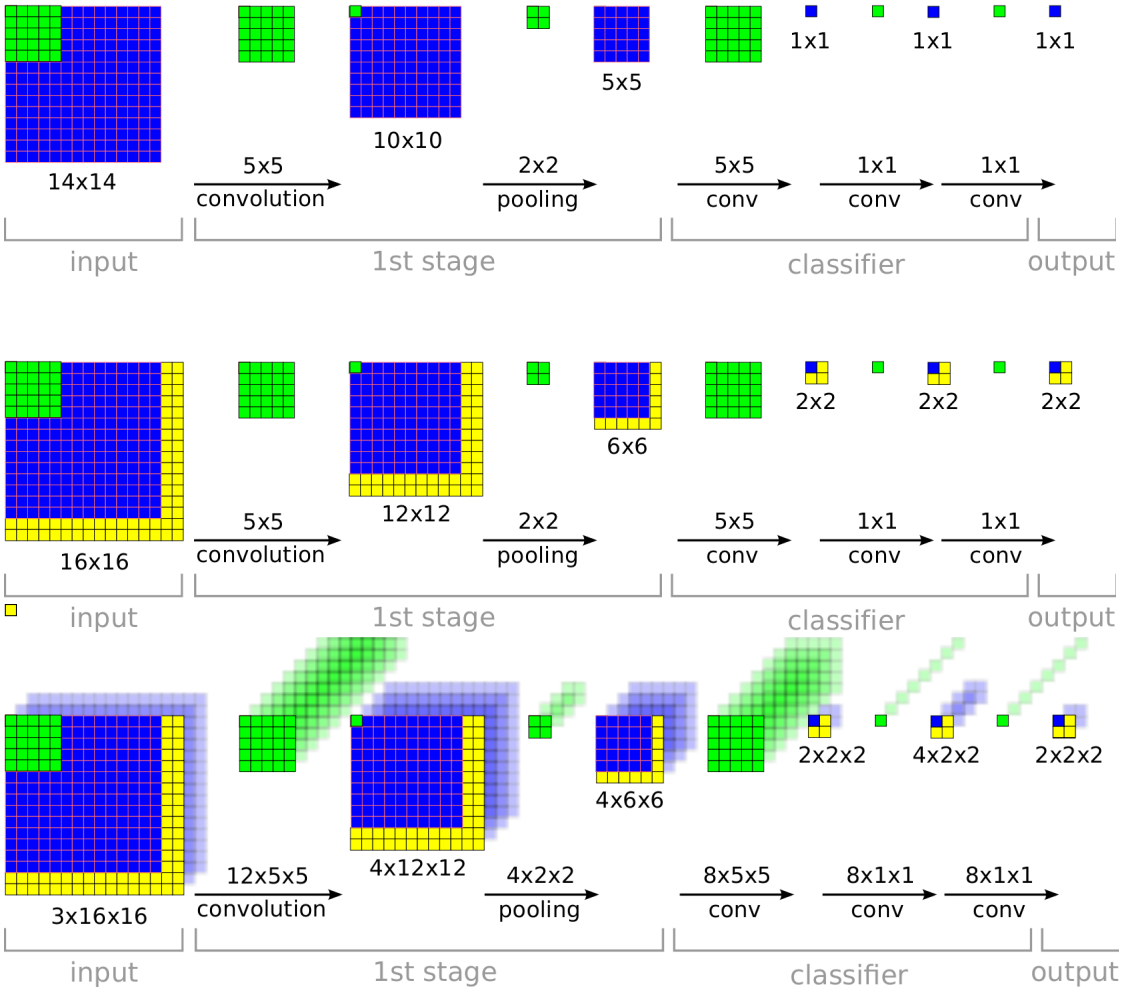

Figure 4.7: **The efficiency of ConvNets for detection.** During training, a ConvNet produces only 1 spatial output (top). But when applied densely over a bigger input image, it produces a spatial output map, e.g. 2x2 (middle). Since all layers of a ConvNet are applied convolutionally, only the yellow region needs to be recomputed when comparing to the top diagram. The feature dimension was removed for simplicity in the top and middle diagrams and added to the bottom diagram.

location provides a score of confidence that an object of class c is present (though not necessarily fully contained) in the corresponding field of view. Thus we can assign to each bounding box a confidence.

Localization within a view is performed by training a regressor on top of the classification network features, described in Section 3.1, to predict the bounding box of the object.

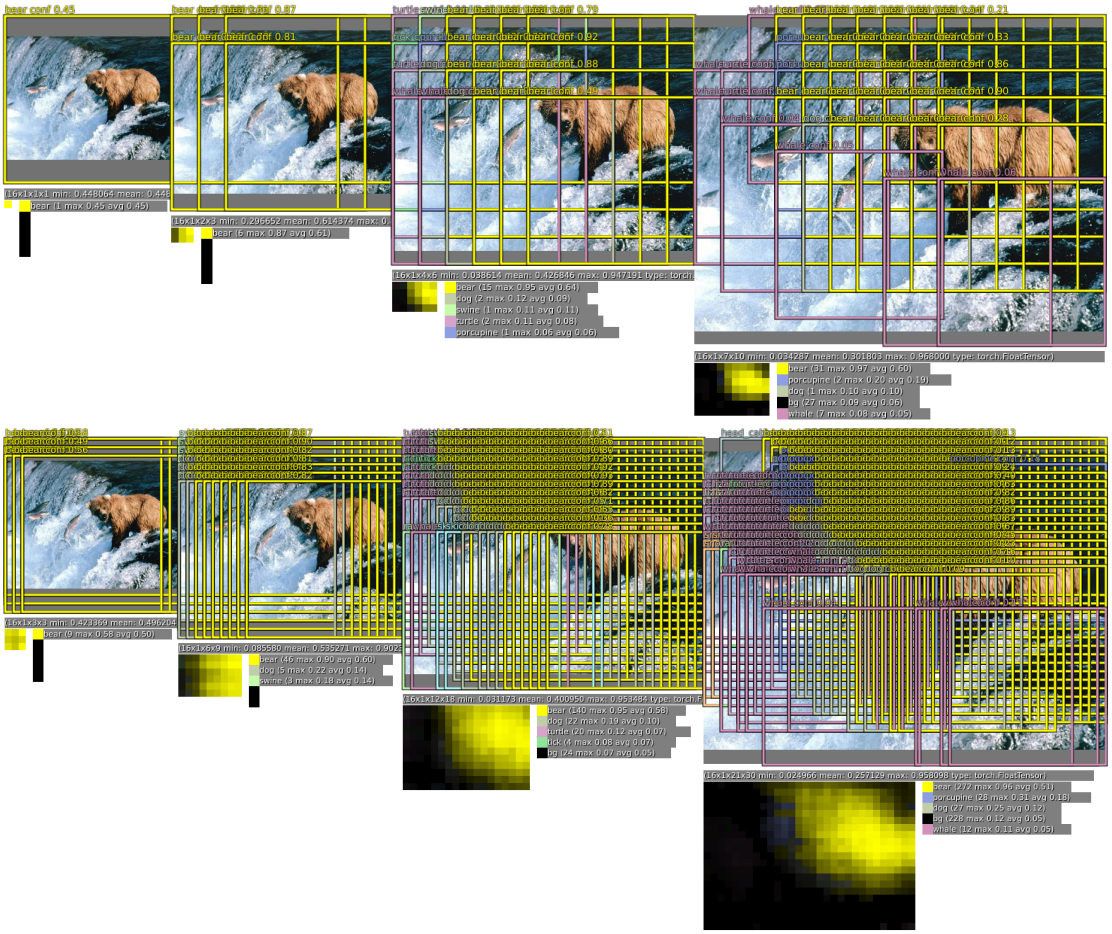

Figure 4.8: **Localization/Detection pipeline: fine stride sliding window.** The raw classifier/detector outputs a class and a confidence for each location (top). The resolution of these predictions can be increased using the method described in section 3.1.3 (bottom).

4.3.2 Regressor Training

The regression network takes as input the pooled feature maps from layer 5. It has 2 fully-connected hidden layers of size 4096 and 1024 channels, respectively. The output

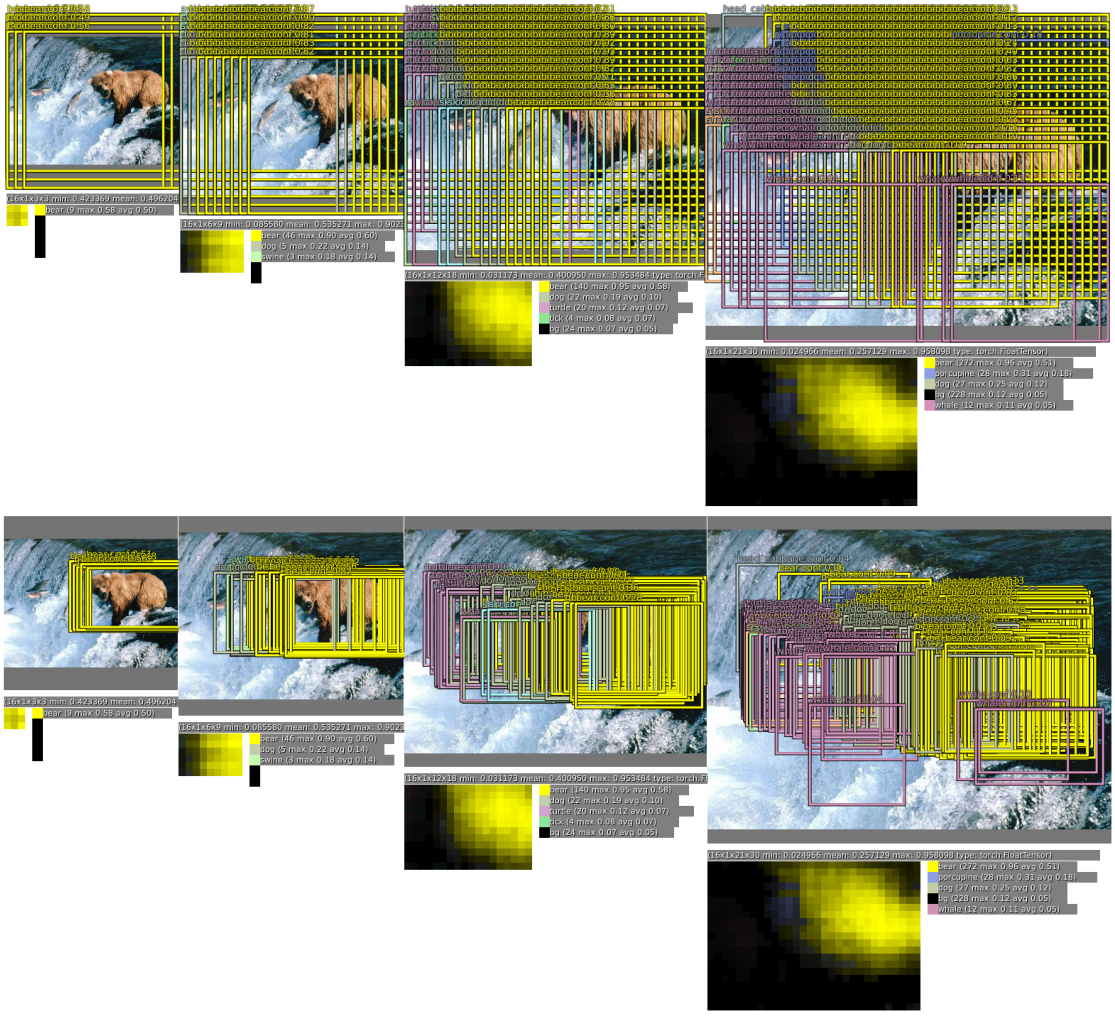

Figure 4.9: **Localization/Detection pipeline: fusing classification and localization predictions.** Top: the raw classifier/detector outputs a class and a confidence for each location using the fine striding approach (Figure 4.8). The regression then predicts the location scale of the object with respect to each window and assigns the predicted classes to each predicted bounding box (bottom).

layer is different for each class, and has 4 units which specify the coordinates for the bounding box edges. As with classification, there are (3x3) copies throughout, resulting from the Δ_x, Δ_y shifts. The architecture is shown in Figure 4.12.

We fix the feature extraction layers (1-5) from the classification network and train the regression network using an ℓ_2 loss between the predicted and true bounding box for

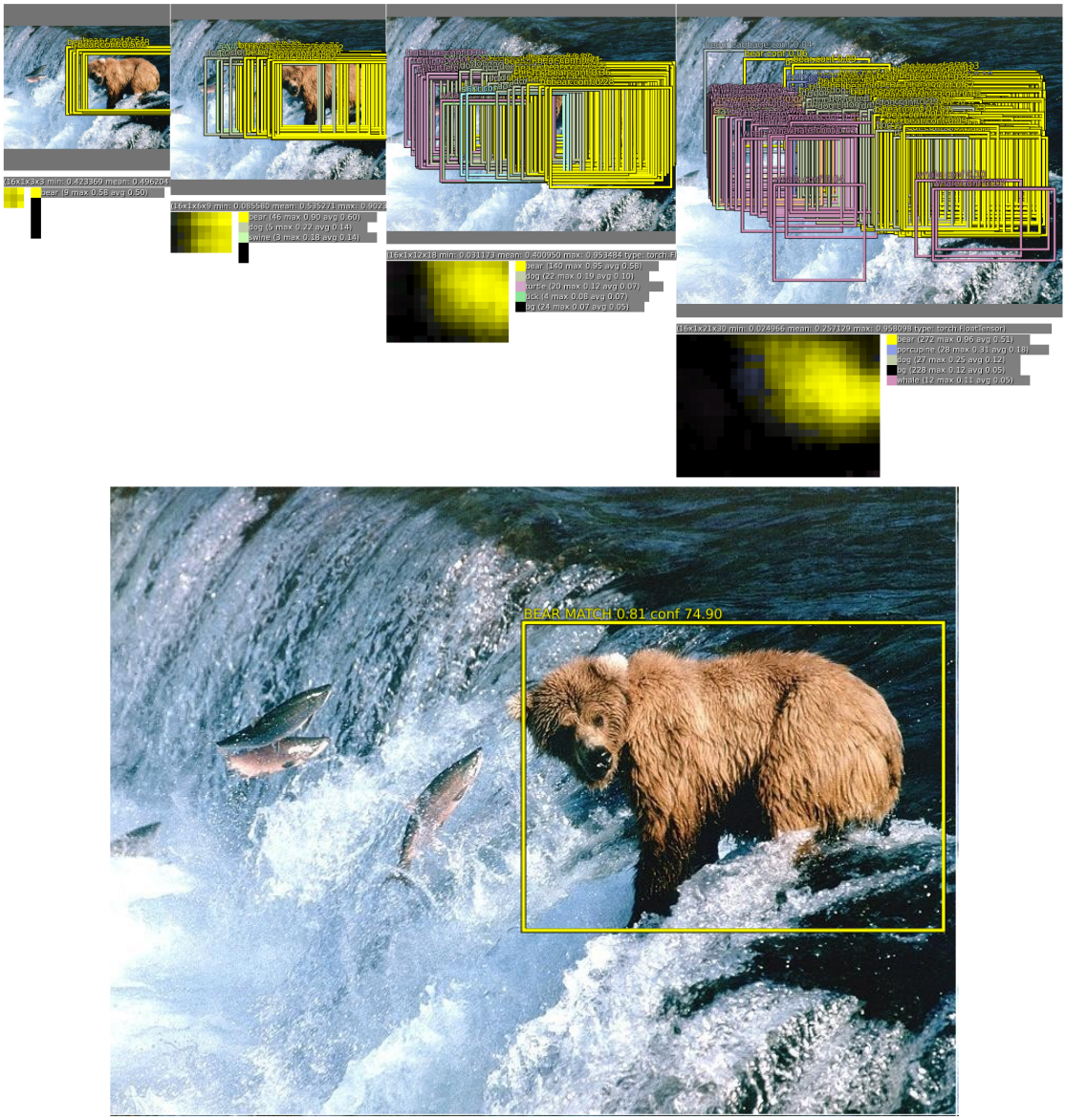

Figure 4.10: **Localization/Detection pipeline: fusing bounding boxes.** Once labels have been assigned to predicted bounding boxes (Figure 4.9), the boxes with high match are fused and their confidence is increased, reducing the set of bounding boxes to only a few (bottom). The ones with confidence lower than a certain threshold are dropped. Here, we obtain a single high confidence (74.9) bounding box (the initial individual bounding boxes have a confidence range of [0, 1]).

each example. The final regressor layer is class-specific, having 1000 different versions, one for each class. We train this network using the same set of scales as described in

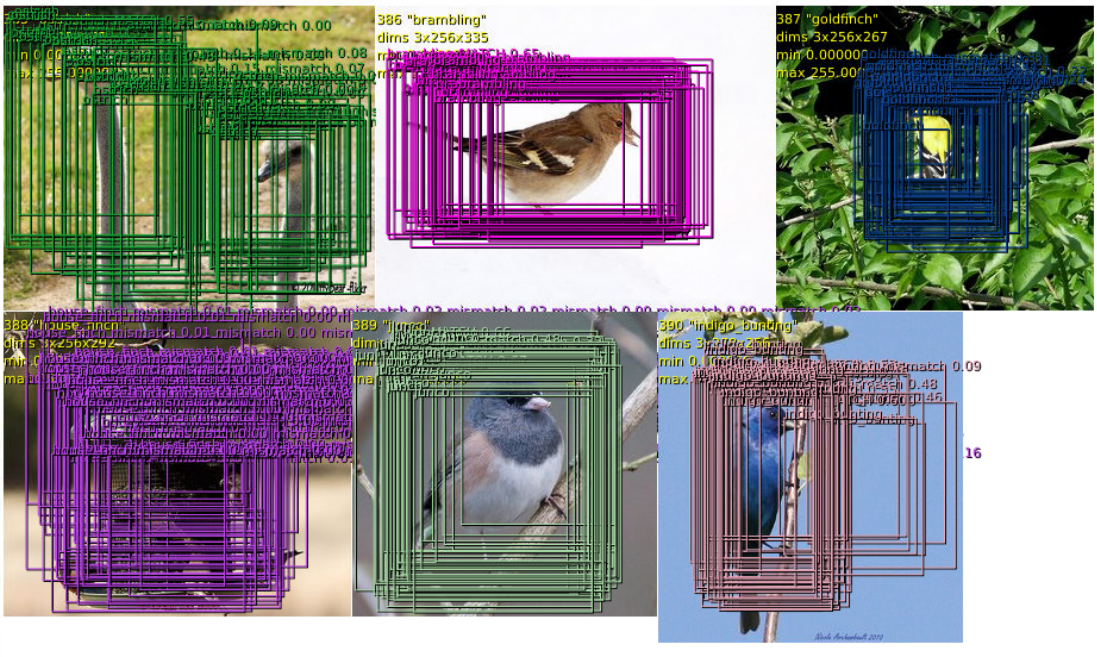

Figure 4.11: **Examples of bounding boxes produced by the regression network**, before being combined into final predictions. The examples shown here are at a single scale. Predictions may be more optimal at other scales depending on the objects. Here, most of the bounding boxes which are initially organized as a grid, converge to a single location and scale. This indicates that the network is very confident in the location of the object, as opposed to being spread out randomly. The top left image shows that it can also correctly identify multiple location if several objects are present. The various aspect ratios of the predicted bounding boxes shows that the network is able to cope with various object poses.

Section 3.1. We compare the prediction of the regressor net at each spatial location with the ground-truth bounding box, shifted into the frame of reference of the regressor's translation offset within the convolution (see Figure 4.12). However, we do not train the regressor on bounding boxes with less than 50% overlap with the input field of view: since the object is mostly outside of these locations, it will be better handled by regression windows that do contain the object.

Training the regressors in a multi-scale manner is important for the across-scale prediction combination. Training on a single scale will perform well on that scale and still

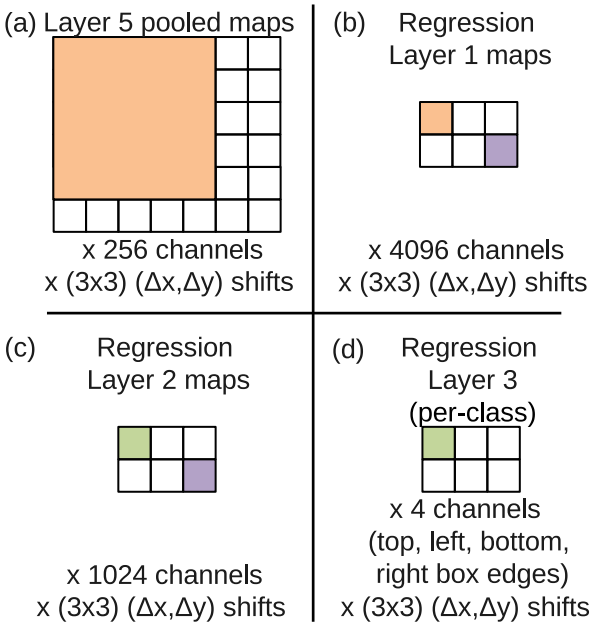

Figure 4.12: Application of the regression network to layer 5 features, at scale 2, for example. (a) The input to the regressor at this scale are 6x7 pixels spatially by 256 channels for each of the (3x3) Δ_x, Δ_y shifts. (b) Each unit in the 1st layer of the regression net is connected to a 5x5 spatial neighborhood in the layer 5 maps, as well as all 256 channels. Shifting the 5x5 neighborhood around results in a map of 2x3 spatial extent, for each of the 4096 channels in the layer, and for each of the (3x3) Δ_x, Δ_y shifts. (c) The 2nd regression layer has 1024 units and is fully connected (i.e. the purple element only connects to the purple element in (b), across all 4096 channels). (d) The output of the regression network is a 4-vector (specifying the edges of the bounding box) for each location in the 2x3 map, and for each of the (3x3) Δ_x, Δ_y shifts.

perform reasonably on other scales. However training multi-scale will make predictions match correctly across scale and exponentially increase the confidence of the merged predictions. In turn, this allows the network to perform well with a few scales only rather than many scales as is typically the case in detection. The typical ratio from one scale to another in pedestrian detection [32] is about 1.05 to 1.1, here however we use a large ratio of approximately 1.4 (this number differs for each scale since dimensions are adjusted to fit exactly the stride of our network) which allows us to run our system faster.

4.3.3 Combining Predictions

We combine the individual predictions (see Figure 4.11) via a greedy merge strategy applied to the regressor bounding boxes, using the following algorithm.

(a) Assign to C_s the set of classes in the top k for each scale $s \in 1 \ldots 6$, found by taking the maximum detection class outputs across spatial locations for that scale.

(b) Assign to B_s the set of bounding boxes predicted by the regressor network for each class in C_s, across all spatial locations at scale s.

(c) Assign $B \leftarrow \bigcup_s B_s$

(d) Repeat merging until done:

(e) $(b_1^*, b_2^*) = \text{argmin}_{b_1 \neq b_2 \in B} \texttt{match\_score}(\texttt{b}_1, \texttt{b}_2)$

(f) If $\texttt{match\_score}(b_1^*, b_2^*) > t$, stop.

(g) Otherwise, set $B \leftarrow B \backslash \{b_1^*, b_2^*\} \cup \texttt{box\_merge}(b_1^*, b_2^*)$

In the above, we compute `match_score` using the sum of the distances between centers of the two bounding boxes and the intersection area of the boxes. `box_merge` computes the average of the bounding boxes' coordinates.

The final prediction is given by taking the merged bounding boxes with maximum class scores. This is computed by cumulatively adding the detection class outputs associated with the input windows from which each bounding box was predicted. See Figure 4.10 for an example of bounding boxes merged into a single high-confidence bounding box. In that example, some *turtle* and *whale* bounding boxes appear in the intermediate multi-scale steps, but disappear in the final detection image. Not only do these bounding boxes have low classification confidence (at most 0.11 and 0.12 respectively), their collection is not as coherent as the *bear* bounding boxes to get a significant confidence boost. The *bear* boxes however have a strong confidence (approximately 0.5 average confidence

per scale) and high matching scores. Hence after merging, many *bear* bounding boxes fused into a single very high confidence box, while false positives disappear below the detection threshold due their lack of bounding box coherence and confidence. This analysis suggests that our approach is naturally more robust to false positives coming from the pure-classification model than traditional non-maximum suppression, by rewarding bounding box coherence.

4.3.4 Experiments

We apply our network to the Imagenet 2012 validation set, using the localization criterion specified for the competition. The results are shown in Figure 4.13. Training and testing data are the same for 2012 and 2013 competitions ; results are reported for both in Figure 4.14. Our method is the winner of the 2013 competition with 29.9% error.

Our multiscale and multi-view approach was critical to obtaining good performance, as can be seen in Figure 4.13: using only a single centered crop, our regressor network achieves an error rate of 40%. By combining regressor predictions from all spatial locations at two scales, we achieve a vastly better error rate of 31.5%. Adding a third and fourth scale further improves performance to 30.0% error.

Using a different top layer for each class in the regressor network for each class (Per-Class Regressor (PCR) in Figure 4.13) surprisingly did not outperform using only a single network shared among all classes (44.1% vs. 31.3%). This may be because there are relatively few examples per class annotated with bounding boxes in the training set, while the network has 1000 times more top-layer parameters, resulting in insufficient training. It is possible this approach may be improved by sharing parameters only among similar classes (e.g. training one network for all classes of dogs, another for vehicles, etc.).

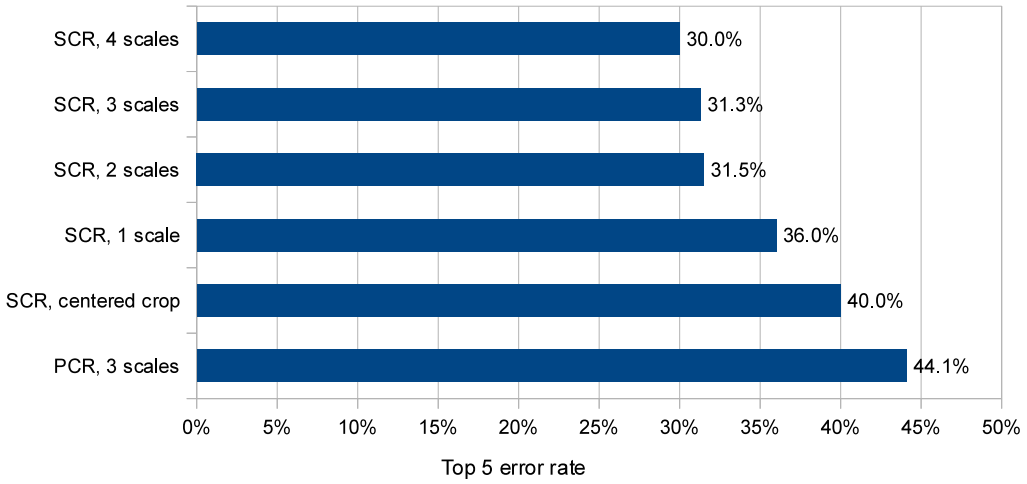

Figure 4.13: **Localization experiments on ILSVRC12 validation set.** We experiment with different number of scales and with the use of single-class regression (SCR) or per-class regression (PCR).

4.4 Object Detection

Detection training is similar to classification training but in a spatial manner. Multiple locations of an image may be trained simultaneously. Since the model is convolutional, all weights are shared among all locations. The main difference with the localization task, is the necessity to predict a background class when no object is present. Traditionally, negative examples are initially taken at random for training. Then the most offending negative errors are added to the training set in bootstrapping passes. Independent bootstrapping passes render training complicated by requiring manual intervention or convoluted programming. Moreover, by decoupling the negative samples extraction from the training there is a risk of mistakenly inducing subtle differences between the bootstrapping and training times. Additionally, the size of bootstrapping passes needs to be tuned to make sure training does not overfit on a small set. To circumvent all these problems, we perform negative training on the fly, by selecting a few

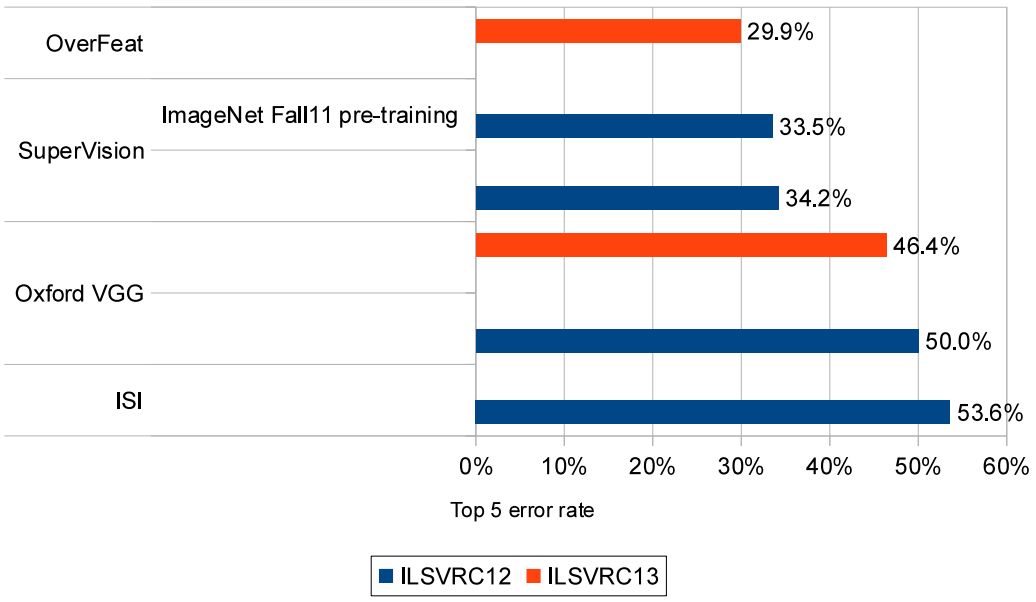

Figure 4.14: **ILSVRC12 and ILSVRC13 competitions results (test set).** Our entry is the winner of the ILSVRC13 localization competition with 29.9% error (top 5). Note that training and testing data is the same for both years. The OverFeat entry uses 4 scales and a single-class regression approach.

interesting negative examples per image such as random ones or most offending ones. This approach is more computationally expensive, but renders the procedure much simpler. And since the feature extraction is initially trained with the classification task, the detection fine-tuning is not as long anyway.

In Figure 4.15, we report the results of the ILSVRC 2013 competition where our detection system ranked 3rd with 19.4% mean average precision (mAP). We later established a new detection state of the art with 24.3% mAP. Note that there is a large gap between the top 3 methods and other teams (the 4th method yields 11.5% mAP). Additionally, our approach is considerably different from the top 2 other systems which use an initial segmentation step to reduce candidate windows from approximately 200,000 to 2,000. This technique speeds up inference and substantially reduces the number of potential false positives. [43, 44] suggest that detection accuracy drops when using dense

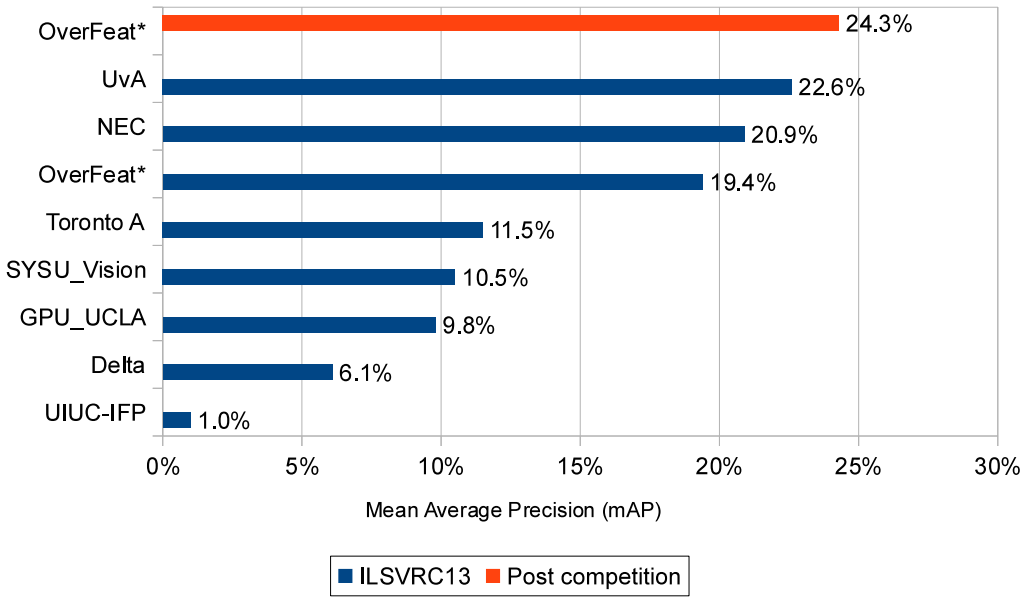

Figure 4.15: **ILSVRC13 test set Detection results.** During the competition, UvA ranked first with 22.6% mAP. In post competition work, we establish a new state of the art with 24.3% mAP. Systems marked with * were pre-trained with the ILSVRC12 classification data.

sliding window as opposed to selective search which discards unlikely object locations hence reducing false positives. Combined with our method, we may observe similar improvements as seen here between traditional dense methods and segmentation based methods. It should also be noted that we did not fine tune on the detection validation set as NEC and UvA did. The validation and test set distributions differ significantly enough from the training set that this alone improves results by approximately 1 point. The improvement between the two OverFeat results in Figure 4.15 are due to longer training times and the use of context, i.e. each scale also uses lower resolution scales as input.

4.5 Speech Localization

We hypothesize that our localization approach described earlier can generalize to other modalities. In particular in speech, sub-phone classification can benefit from decoupling the classification and localization steps. Currently, a speech signal is sliced into 10 milliseconds frames and the acoustic model must classify each of these frames into one of many sub-phone classes (3,000 for Cantonese and 4,500 for Vietnamese in the Babel datasets). Many of these sub-phones have the same class as their neighbors over some period of time. However some classes appear in single sub-phones with neighbors of different classes. Over a context window of 40 frames, this turns this problem into a difficult localization problem because the answers must be very precise along the temporal dimension. An analogy to the vision localization problem would be trying to train a network to fire precisely at the exact location of an object and predict a different class when moving the network window by one pixel away. To achieve this, one would have to give up the invariance power brought by the pooling layers. We hypothesize that this explains why temporal subsampling was not effective in the case of [24].

Instead of approaching the speech problem from a very precise localization problem, we propose to relax the localization contraint and assign that task to a separate branch of the network trained specifically for that. This would allow recovery of the invariance of temporal pooling, thus increasing the pure classification results. Just like the localization network described in Section 4.3, the existing ConvNet pipeline can be branched out before the classifier and fed to a regression network that predicts the temporal location of the sub-phone. It is an even easier problem than object detection (which must predict a 4-dimensional output), because it only requires prediction a 1-dimensional output. In addition to this decoupling to relax the problem, the accumulation aspect of our object localization system will boost performance by voting of many different views of the same sub-phone. Indeed, the network is also applied in a sliding window fashion, hence providing many translated views of the same locations. The accumulation of both classi-

fication and localization evidence will provide robustness to the sub-phone classification problem. Due to time contraints however, this approach was not experimented with on speech data at the time of writing.

Chapter 5

Conclusions and Discussion

This thesis advances the research on object detection significantly by breaking the record on one of the most challenging detection datasets to date using a novel approach to localization. After improving acoustic feature learning, we hypothesize that significant improvements can also be gained by applying the same localization method to speech recognition.

First, we explored how to learn good features using Convolutional Networks for computer vision and speech recognition. Second, we applied traditional and novel approaches to object detection using ConvNets. We showed how ConvNets can be used effectively not only for classification tasks, but also for more challenging ones such as localization and detection in an unified manner and for different modalities. This was demonstrated on a number of popular benchmarks on which we established several records, including one of the first super-human vision classification records.

Our approach could be improved in numerous ways: (i) for localization, we are not currently back-propping through the whole network; doing so is likely to improve performance. (ii) we are using ℓ_2 loss, rather than directly optimizing the intersection-over-union (IOU) criterion on which performance is measured. Swapping the loss to this should be possible since IOU is still differentiable, provided there is some overlap. (iii)

alternate parameterizations of the bounding box may help to decorrelate the outputs, which will aid network training. Additionally, some of the results presented were not all based on fully trained ConvNets because of time constraints. The classification, localization and detection results are expected to improve over time. Because of time constraints as well, we could not conduct experiments to analyze the individual benefits of all components proposed in this thesis. A thorough analysis will help determine the most beneficial pieces. Finally, it remains to apply our approach to object localization to improve phone classification for speech recognition.

Bibliography

[1] Ciresan, D. C, Meier, U, Masci, J, and Schmidhuber, J. A committee of neural networks for traffic sign classification. In *International Joint Conference on Neural Networks*, pages 1918–1921, 2011. 6

[2] Dollár, P, Tu, Z, Perona, P, and Belongie, S. Integral channel features. In *BMVC 2009, London, England.* 8

[3] Dalal, N and Triggs, B. Histograms of oriented gradients for human detection. In Schmid, C, Soatto, S, and Tomasi, C, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005. 8, 44, 46

[4] Felzenszwalb, P, Girshick, R, McAllester, D, and Ramanan, D. Object detection with discriminatively trained part based models. In *PAMI 2010.* 8

[5] Schwartz, W. R, Kembhavi, A, Harwood, D, and Davis, L. S. Human detection using partial least squares analysis. In *Computer Vision, 2009 IEEE 12th International Conference on*, 29 2009. 8

[6] Walk, S, Majer, N, Schindler, K, and Schiele, B. New features and insights for pedestrian detection. In *CVPR 2010, San Francisco, California.* 8

[7] Maji, S, Berg, A. C, and Malik, J. Classification using intersection kernel support vector machines is efficient. volume 0, pages 1–8, Los Alamitos, CA, USA, 2008. IEEE Computer Society. 8

[8] Dollár, P, Appel, R, and Kienzle, W. Crosstalk cascades for frame-rate pedestrian detection. 8

[9] LeCun, Y, Bottou, L, Orr, G, and Muller, K. Efficient backprop. In Orr, G and K., M, editors, *Neural Networks: Tricks of the trade*. Springer, 1998. 8, 32, 46

[10] Hinton, G. E and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 8

[11] Bengio, Y, Lamblin, P, Popovici, D, and Larochelle, H. Greedy layer-wise training of deep networks. In Schölkopf, B, Platt, J, and Hoffman, T, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press, 2007. 8

[12] Ranzato, M, Boureau, Y, and LeCun, Y. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems (NIPS 2007)*, volume 20, 2007. 8

[13] Jarrett, K, Kavukcuoglu, K, Ranzato, M, and LeCun, Y. What is the best multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision (ICCV'09)*. IEEE, 2009. 8, 9, 30

[14] Kavukcuoglu, K, Sermanet, P, Boureau, Y, Gregor, K, Mathieu, M, and LeCun, Y. Learning convolutional feature hierachies for visual recognition. In *Advances in Neural Information Processing Systems (NIPS 2010)*, 2010. 8, 9, 30, 48, 50

[15] LeCun, Y, Bottou, L, Bengio, Y, and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. 9, 26

[16] LeCun, Y, Boser, B, Denker, J. S, Henderson, D, Howard, R. E, Hubbard, W, and Jackel, L. D. Handwritten digit recognition with a back-propagation network. In Touretzky, D, editor, *Advances in Neural Information Processing Systems (NIPS 1989)*, volume 2, Denver, CO, 1990. Morgan Kaufman. 9

[17] Sermanet, P, Chintala, S, and LeCun, Y. Convolutional neural networks applied to house numbers digit classification. In *International Conference on Pattern Recognition (ICPR 2012)*, 2012. 9, 27, 29, 36

[18] LeCun, Y, Huang, F.-J, and Bottou, L. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of CVPR'04*. IEEE Press, 2004. 9

[19] Ciresan, D. C, Meier, J, and Schmidhuber, J. Multi-column deep neural networks for image classification. In *CVPR*, 2012. 9

[20] Sermanet, P and LeCun, Y. Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN'11)*, 2011. 9, 27, 30, 38

[21] Krizhevsky, A, Sutskever, I, and Hinton, G. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 9, 11, 12, 13, 14, 16, 19, 20, 38

[22] Deng, J, Dong, W, Socher, R, Li, L.-J, Li, K, and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 9, 12

[23] Hinton, G, Deng, L, Yu, D, Dahl, G. E, Mohamed, A.-r, Jaitly, N, Senior, A, Vanhoucke, V, Nguyen, P, Sainath, T. N, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012. 9

[24] Sainath, T. N, Mohamed, A.-r, Kingsbury, B, and Ramabhadran, B. Deep convolutional neural networks for lvcsr. In *Acoustics, Speech and Signal Processing*

(ICASSP), 2013 IEEE International Conference on, pages 8614–8618. IEEE, 2013. 10, 22, 24, 73

[25] Sainath, T. N, Kingsbury, B, Mohamed, A.-r, Dahl, G. E, Saon, G, Soltau, H, Beran, T, Aravkin, A. Y, and Ramabhadran, B. Improvements to deep convolutional neural networks for lvcsr. *arXiv preprint arXiv:1309.1501*, 2013. 10, 27

[26] Matan, O, Bromley, J, Burges, C, Denker, J, Jackel, L, LeCun, Y, Pednault, E, Satterfield, W, Stenard, C, and Thompson, T. Reading handwritten digits: A zip code recognition system. *IEEE Computer*, 25(7):59–63, July 1992. 10

[27] Vaillant, R, Monrocq, C, and LeCun, Y. Original approach for the localisation of objects in images. *IEE Proc on Vision, Image, and Signal Processing*, 141(4):245–250, August 1994. 10

[28] Nowlan, S and Platt, J. A convolutional neural network hand tracker. pages 901–908, San Mateo, CA, 1995. Morgan Kaufmann. 10

[29] Delakis, M and Garcia, C. Text detection with convolutional neural networks. In *International Conference on Computer Vision Theory and Applications (VISAPP 2008)*, 2008. 10

[30] Garcia, C and Delakis, M. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. 10

[31] Osadchy, M, LeCun, Y, and Miller, M. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8:1197–1215, May 2007. 10

[32] Sermanet, P, Kavukcuoglu, K, Chintala, S, and LeCun, Y. Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'13)*. IEEE, June 2013. 10, 67

[33] Taylor, G, Fergus, R, Williams, G, Spiro, I, and Bregler, C. Pose-sensitive embedding by nonlinear nca regression. In *NIPS*, 2011. 10

[34] Taylor, G, Spiro, I, Bregler, C, and Fergus, R. Learning invarance through imitation. In *CVPR*, 2011. 10

[35] Hinton, G. E, Krizhevsky, A, and Wang, S. D. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 44–51. Springer Berlin Heidelberg, 2011. 10

[36] Jain, V, Murray, J. F, Roth, F, Turaga, S, Zhigulin, V, Briggman, K, Helmstaedter, M, Denk, W, and Seung, H. S. Supervised learning of image restoration with convolutional networks. In *ICCV'07*. 11

[37] Ning, F, Delhomme, D, LeCun, Y, Piano, F, Bottou, L, and Barbano, P. Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371, September 2005. Special issue on Molecular and Cellular Bioimaging. 11

[38] Hadsell, R, Sermanet, P, Scoffier, M, Erkan, A, Kavackuoglu, K, Muller, U, and LeCun, Y. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, February 2009. 11

[39] Farabet, C, Couprie, C, Najman, L, and LeCun, Y. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013. in press. 11

[40] Manen, S, Guillaumin, M, and Van Gool, L. Prime object proposals with randomized prims algorithm. In *International Conference on Computer Vision (ICCV)*, 2013. 11

[41] Carreira, J and Sminchisescu, C. Constrained parametric min-cuts for automatic object segmentation, release 1. http://sminchisescu.ins.uni-bonn.de/code/cpmc/. 11

[42] Endres, I and Hoiem, D. Category independent object proposals. In *Computer Vision–ECCV 2010*, pages 575–588. Springer, 2010. 11

[43] Uijlings, J. R. R, van de Sande, K. E. A, Gevers, T, and Smeulders, A. W. M. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013. 11, 71

[44] Carreira, J, Li, F, and Sminchisescu, C. Object recognition by sequential figure-ground ranking. *International journal of computer vision*, 98(3):243–262, 2012. 11, 71

[45] Girshick, R, Donahue, J, Darrell, T, and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013. 11

[46] Netzer, Y, Wang, T, Coates, A, Bissacco, A, Wu, B, and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011. 13, 29, 44

[47] Stallkamp, J, Schlipsing, M, Salmen, J, and Igel, C. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *submitted to International Joint Conference on Neural Networks*, 2011. 14

[48] Hinton, G, Srivastave, N, Krizhevsky, A, Sutskever, I, and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, 2012. 13, 38

[49] Giusti, A, Ciresan, D. C, Masci, J, Gambardella, L. M, and Schmidhuber, J. Fast image scanning with deep max-pooling convolutional neural networks. In *International Conference on Image Processing (ICIP)*, 2013. 17, 19

[50] Tyagi, V and Wellekens, C. On desensitizing the mel-cepstrum to spurious spectral components for robust speech recognition. In *Proc. ICASSP*, volume 5, pages 529–532, 2005. 21

[51] Palaz, D, Collobert, R, and Magimai. -Doss, M. End-to-end Phoneme Sequence Recognition using Convolutional Neural Networks. *ArXiv e-prints, at NIPS Deep Learning Workshop, 2013*, December 2013. 21

[52] Krizhevsky, A, Sutskever, I, and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NIPS 2012: Neural Information Processing Systems, Lake Tahoe, Nevada*. 26

[53] Fan, J, Xu, W, Wu, Y, and Gong, Y. Human tracking using convolutional neural networks. *Neural Networks, IEEE Transactions on*, 21(10):1610 –1623, 2010. 26, 27

[54] Sermanet, P, Kavukcuoglu, K, and LeCun, Y. Traffic signs and pedestrians vision with multi-scale convolutional networks. In *Snowbird Machine Learning Workshop, 2011*. 27

[55] Simoncelli, E. P and Heeger, D. J. A model of neuronal responses in visual area mt, 1997. 27

[56] Hyvrinen, A and Kster, U. Complex cell pooling and the statistics of natural images. In *Computation in Neural Systems,*, 2005. 27

[57] Kavukcuoglu, K, Ranzato, M, Fergus, R, and LeCun, Y. Learning invariant features through topographic filter maps. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'09)*. IEEE, 2009. 28

[58] Yang, J, Yu, K, Gong, Y, and Huang, T. Linear spatial pyramid matching using sparse coding for image classification. In *in IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 28

[59] Boureau, Y, Ponce, J, and LeCun, Y. A theoretical analysis of feature pooling in vision algorithms. In *Proc. International Conference on Machine learning (ICML'10)*, 2010. 28

[60] Ciresan, D. C, Meier, U, Gambardella, L. M, and Schmidhuber, J. Deep big multilayer perceptrons for digit recognition. *Neural Networks Tricks of the Trade*, 1:581–598, 2012. 30

[61] Stallkamp, J, Schlipsing, M, Salmen, J, and Igel, C. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 2012. 30

[62] Nair, V and Hinton, G. Rectified linear units improve restricted boltzmann machines. 2010. 32

[63] Eigen, D, Rolfe, J, Fergus, R, and LeCun, Y. Understanding deep architectures using a recursive convolutional network. *arXiv preprint arXiv:1312.1847*, 2013. 37

[64] Goodfellow, I. J, Warde-Farley, D, Mirza, M, Courville, A, and Bengio, Y. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013. 37, 38

[65] Cireşan, D. C, Meier, U, Masci, J, Gambardella, L. M, and Schmidhuber, J. High-performance neural networks for visual object classification. *arXiv preprint arXiv:1102.0183*, 2011. 37

[66] Claudiu Ciresan, D, Meier, U, Gambardella, L. M, and Schmidhuber, J. Deep big simple neural nets excel on handwritten digit recognition. *arXiv preprint arXiv:1003.0358*, 2010. 38

[67] Le, Q. V, Ranzato, M, Monga, R, Devin, M, Chen, K, Corrado, G. S, Dean, J, and Ng, A. Y. Building high-level features using large scale unsupervised learning. *arXiv preprint arXiv:1112.6209*, 2011. 38

[68] Yu, D and Seltzer, M. L. Improved bottleneck features using pretrained deep neural networks. 2011. 39

[69] Hinton, G. E and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 39

[70] Saxe, A, Koh, P. W, Chen, Z, Bhand, M, Suresh, B, and Ng, A. In *Adv. Neural Information Processing Systems (NIPS*10), Workshop on Deep Learning and Unsupervised Feature Learning.*, 2010. 39

[71] Dollár, P, Wojek, C, Schiele, B, and Perona, P. Pedestrian detection: A benchmark. In *CVPR'09*. IEEE, June 2009. 44, 45, 52, 53, 58

[72] Aharon, M, Elad, M, and Bruckstein, A. M. K-SVD and its non-negative variant for dictionary design. In Papadakis, M, Laine, A. F, and Unser, M. A, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5914, pages 327–339, August 2005. 47, 48

[73] Daubechies, I, Defrise, M, and De Mol, C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004. 47, 49

[74] Beck, A and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, 2009. 47, 49, 50

[75] Li, Y and Osher, S. Coordinate Descent Optimization for l1 Minimization with Application to Compressed Sensing; a Greedy Algorithm. *CAM Report*, pages 09–17. 47

[76] Olshausen, B. A and Field, D. J. Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997. 48

[77] Mairal, J, Bach, F, Ponce, J, Sapiro, G, and Zisserman, A. Discriminative learned dictionaries for local image analysis. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008. 48

[78] Kavukcuoglu, K, Ranzato, M, and LeCun, Y. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU, 2008. Tech Report CBLL-TR-2008-12-01. 48

[79] Zeiler, M, Krishnan, D, Taylor, G, and Fergus, R. Deconvolutional Networks. In *CVPR'10*. IEEE, 2010. 48